NASA Contractor Report 198460

An Assessment of IMPAC—Integrated Methodology for Propulsion and Airframe Controls

G.P. Walker, E.A. Wagner, and D.S. Bodden
Lockheed Martin Tactical Aircraft Systems
Fort Worth, Texas

April 1996

Prepared for
Lewis Research Center
Under Contract NAS1–19000

National Aeronautics and
Space Administration

Table of Contents

Table of Contents, cont'd

Table of Contents, cont'd

List of Illustrations

List of Illustrations, cont'd

Nomenclature

Acronyms and Symbols

a/f	airframe
a_i	variable in K^{lead}
A	In standard H_∞ problem, x to dx/dt state equation matrix
A78	ventral nozzle area, in^2
$A78_{max}$	maximum ventral nozzle area, in^2
A8	aft nozzle area, in^2
$A8_{max}$	maximum aft nozzle area, in^2
ANG79	ventral nozzle vectoring angle, deg.
ANG8	aft nozzle vectoring angle, deg.
AQR	pitch RCS area, in^2
b_i	variable in K^{lead}
B_1	In standard H_∞ problem, w to x–dot state equation matrix
B_2	In standard H_∞ problem, u to x–dot state equation matrix
C	control transmission function
C_1	In standard H_∞ problem, x to z state equation matrix
D_{11}	In standard H_∞ problem w to z state equation matrix
D_{12}	In standard H_∞ problem, u to z state equation matrix
D_{21}	In standard H_∞ problem, w to y state equation matrix
D_{22}	In standard H_∞ problem, u to y state equation matrix
DMICS	Design Methods for Integrated Control Systems program
$\underline{e}$	errors in tracking commanded controlled variables
$\underline{e}_a$	error in tracking $\underline{z}_a$, airframe controlled variables
$\underline{e}_e$	error in tracking $\underline{z}_e$, engine controlled variables

Nomenclature, cont'd

ETA	ejector butterfly valve angle, deg.
ETA_{max}	maximum ejector butterfly valve angle, deg.
FG9	aft nozzle gross thrust, lb_f
FGE	ejector gross thrust, lb_f
FGV	ventral nozzle gross thrust, lb_f
G	integrated plant transfer function
G_{aa}	transfer function from open–loop airframe controls to airframe controlled variables and measured feedbacks
G_{ae}	transfer function from open–loop engine controls to airframe controlled variables and measured feedbacks
G^a_{ea}	transfer function from $\underline{u}_a$ to $\underline{z}_{ea}$
G^e_{ea}	transfer function from $\underline{u}_e$ to $\underline{z}_{ea}$
G_{ea}	transfer function from open–loop airframe controls to engine controlled variables and measured feedbacks
G_{ee}	transfer function from open–loop engine controls to engine controlled variables and measured feedbacks
GUI	graphical user interface
h	altitude, ft
H	weighted combination of transfer matrices to be minimized in H control design
H_∞	infinity norm of H ($\| \ \|_\infty$ = supremum$_\omega$ (σ_{max}[ ]).)
IMPAC	Integrated Methodology for Propulsion and Airframe Controls
j	complex $\sqrt{-1}$
K	regulator matrix
K^a	partitioned airframe controller
$\widetilde{K}^a(s)$	airframe compensation in partitioned airframe subcontroller K^a

Nomenclature, cont'd

K_{aa}	partition of controller from airframe feedbacks to airframe commands
K_{ae}	partition of controller from engine feedbacks to airframe commands
K^e	partitioned propulsion controller
K^e_e	engine compensation in engine subcontroller K^e
K^e_{ea}	engine–airframe compensation in engine subcontroller K^e
K_{ea}	partition of controller from airframe feedbacks to engine commands
K_{ee}	partition of controller from engine feedbacks to engine commands
K^{lead}	lead compensation in partitioned airframe controller K^a
LMTAS	Lockheed Martin Tactical Aircraft Systems
N2	engine fan speed, rpm
$N2_{max}$	maximum engine fan speed, rpm
N25	high pressure compressor speed, rpm
P	command prefilter
q	pitch rate, rad/s
Qv	$q + 0.3\,\theta$
RCS	reaction control system
s	Laplace transform variable
S	sensitivity function
STOVL	Short Takeoff Vertical Landing
T	complementary sensitivity function
T^{ae}_{ea}	closed–loop frequency response from airframe commands $\underline{z}_a$ to $\underline{z}_{ea}$
$T^{ae}_{ea_i}$	closed–loop frequency response from airframe commands $\underline{z}_a$ to $\underline{z}_{ea_i}$
Tmhpc	high pressure compressor metal temp., °R
Tmhpt	high pressure turbine metal temp., °R
Tmlpt	low pressure turbine metal temp., °R
Tmpc	burner metal temp., °R

Nomenclature, cont'd

u	axial velocity, ft/s
$\underline{u}$	control inputs
$\dot{\underline{u}}$, $\underline{u}$–dot	control input rates
$\underline{u}_a$	airframe subsystem control inputs
$\underline{u}_c$	commanded control inputs
$\underline{u}_e$	engine subsystem control inputs
$\underline{u}_e$–dot	engine subsystem control input rates
$\underline{u}_{e_{max}}$	maximum available engine control inputs
$\dot{\underline{u}}_{e_{max}}$	maximum available engine control input rates
$\underline{u}_{max}$	maximum available control input
$\dot{\underline{u}}_{max}$	maximum available control input rate
V	true airspeed, ft/s
$\dot{V}$	acceleration along flight path, ft/s^2
V_V	$\dot{V} + 0.1\ V$
w	vertical velocity, ft/s
w	In standard H_∞ problem, commands and disturbances
WB3	engine bleed flow demand from RCS, lb_m/s
$WB3_{max}$	maximum engine bleed flow demand from RCS, lb_m/s
WF	fuel flow rate, lb_m/hr
WF_{max}	maximum fuel flow rate, lb_m/hr
W_S	sensitivity weighting matrix
W_T	complementary sensitivity weighting matrix
W_C	control transmission function weighting matrix
W_u	control weighting matrix (part of W_C)
$W_{\dot{u}}$, $W_{u\text{–dot}}$	control rate weighting matrix (part of W_C)

Nomenclature, cont'd

x	In standard H_∞ problem, states
y	In standard H_∞ problem, all feedbacks
$\underline{y}$	measurement feedbacks
$\underline{\hat{y}}$	sensor–compensated measurement feedbacks
$\underline{y}_a$	airframe subsystem measurement feedbacks
$\underline{y}_e$	engine subsystem measurement feedbacks
z	In standard H_∞ problem, all outputs
$\underline{z}$	controlled variables
$\underline{z}_a$	airframe subsystem controlled variables
$\underline{z}_{a,c}$	commanded airframe subsystem controlled variables
$\underline{z}_c$	commanded control variables
$\underline{z}_{c_{max}}$	maximum values of commanded control variables
$\underline{z}_e$	engine subsystem controlled variables
$\underline{z}_{e_c}$	commanded engine subsystem controlled variables
$\underline{z}_{ea}$	interface controlled variables from airframe controller to engine subsystem
$\underline{z}_{ea_c}$	lead–compensated $\underline{z}_{ea_{des}}$
$\underline{z}_{ea_{c_i}}$	individual element of $\underline{z}_{ea_c}$
$\underline{z}_{ea_{des}}$	commanded values of $\underline{z}_{ea}$ before lead compensation
$\underline{z}_{ea_{des,i}}$	individual element of $\underline{z}_{ea_{des}}$
$\underline{z}_{ea_i}$	individual element of $\underline{z}_{ea}$
$\underline{z}_\varepsilon$	controlled variables output by Maneuver Command Generator
$\underline{z}_{na}$	not realizable portion of $\underline{z}_\varepsilon$

Greek Symbols

$\underline{\delta}$	physical control deflections

Nomenclature, cont'd

$\underline{\delta}_{cmd}$	command physical control deflections
δ_e	elevator deflection, deg.
$\underline{\delta}_{trim}$	trim values for physical control deflections
$\underline{\delta}_p$	pilot inputs
$\underline{\delta}^*$	desired generalized control values
$\underline{\delta}_{na}$	not available values for physical controls
γ	flight path angle, deg.
σ	singular value
σ_{max}	maximum singular value
σ_{min}	minimum singular value
ω	frequency variable
ω_{ea_i}	tracking bandwidth for closed–loop $\underline{z}_a$ to $\underline{z}_{ea_i}$ response
θ	pitch attitude, rad

1. SUMMARY

The objective of this project is to transition NASA Lewis Research Center research to industry through development of a software tool that incorporates the IMPAC (Integrated Methodology for Propulsion and Airframe Control) design procedure in a user–friendly environment. This project would leverage Lockheed Martin Tactical Aircraft Systems' (LMTAS) prior practical experience in integrated flight/propulsion and multivariable controls from the STOVL Controls Research and Design Methods for Integrated Control Systems (DMICS) programs.

The end product of this study is a set of software requirements for an IMPAC design tool. To accomplish this the following steps were taken:

1. Exercise the critical steps in the IMPAC methodology in a design example (replicate central parts of a previous NASA–Lewis design effort for an integrated airframe/engine model of the Lockheed Martin Tactical Aircraft Systems E–7D aircraft in transition). MATRIX$_X$® "EXEC" files were created to document each step of the IMPAC procedures. This document includes a listing of these files implementing the IMPAC design process for the E–7D vehicle.

2. Document the design and analysis procedures. Each step of the design and analysis procedures was documented with MATRIX$_X$® output data, plots, and diagrams. For each step, lessons learned were listed along with recommendations that would improve the procedure.

3. Prepare a set of software requirements for a user–friendly tool. The functional characteristics and interfaces were determined from the documented design and analysis procedures. Prototypes for a graphical user interface (GUI) were diagrammed to specify how the tool would interact with the user. Finally, a trade study was performed to assess custom software development versus a shell built around a commercial product (i.e. MATRIX$_X$®, MATLAB®).

After using the IMPAC control design technique with the E–7D aircraft example, our recommendations for the IMPAC methodology are

—Develop guidelines for choosing command variables and control modes.
—Develop guidelines for system partitioning using, e.g., modal, Grammian, relative gain array, singular value, or optimization tools.
—Develop guidelines for choosing H_∞ weighting matrices.
—Develop guidelines for order reduction.
—Consider the use of generalized controls (using a control selector to map commands to physical controls).

The IMPAC design process is a multistep process, and a tool to automate it and provide design guidance could be extremely helpful. Developing such a tool would be straightforward, particularly using one of the integrated control design packages currently available, e.g., MATLAB® or MATRIX$_X$®, and their graphical interface building capabilities. We have diagrammed such a tool from the standpoint of suggested user input and output and predefined functions to implement

distinct steps of the design calculations and drawn some notional menus for an IMPAC tool. We have provided notional menus to implement the IMPAC design process. Context–sensitive "intelligent" help should be available for every menu item available in the IMPAC tool, and we have summarized good ways of implementing help information.

We recommend that an IMPAC tool be developed for both MATLAB® and MATRIX$_X$®. Both are very widely used, and building a custom tool has no significant advantages in cost or usefulness for the control design community.

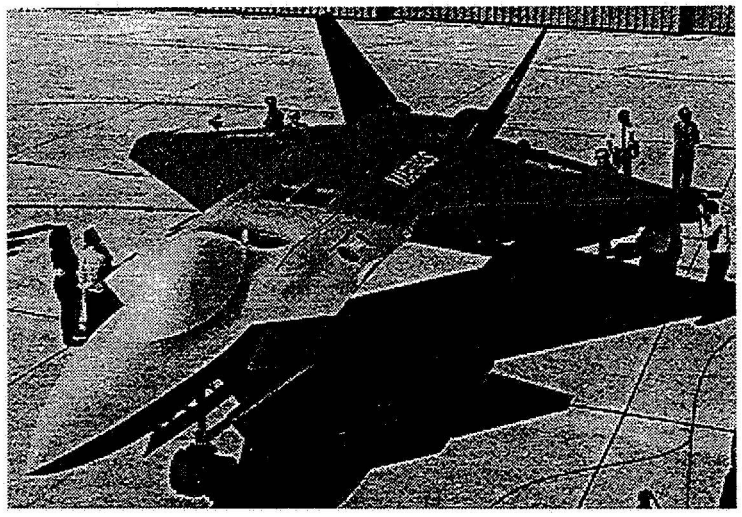
2. INTRODUCTION TO PROGRAM

2.1 Objective

The objective of this project is to transition NASA Lewis Research Center research to industry through development of a software tool that incorporates the IMPAC (Integrated Methodology for Propulsion and Airframe Control) design procedure in a user–friendly environment. This project would leverage Lockheed Martin Tactical Aircraft Systems' (LMTAS) prior practical experience in integrated flight/propulsion and multivariable controls from the STOVL Controls Research and Design Methods for Integrated Control Systems (DMICS) programs.

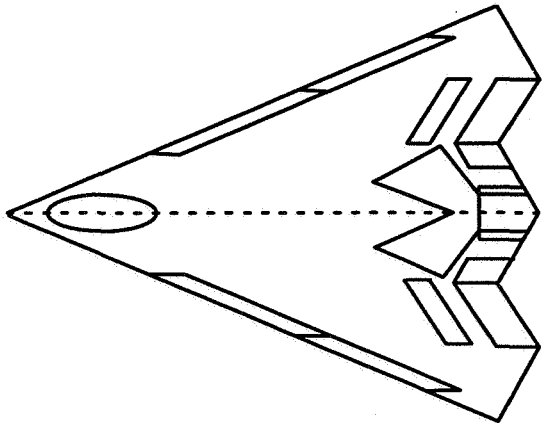
2.2 Background

Aircraft configurations in development today have multiple nonlinear control effectors in each axis (Figure 2.1 and Figure 2.2), strongly suggesting the need for integrated multivariable control.



- Nonlinear Power–Induced Aerodynamics
- Multiple Aero/Propulsion Control Effectors In Each Axis
- Highly Integrated Propulsion System

Figure 2.1 Lockheed Martin JAST STOVL Configuration



- Highly Nonlinear Aero Control Effectors
- Multiple Controls in Each Axis
- Multi–Axis Thrust Vectoring and Pneumatic Vortex Control

Figure 2.2 Lockheed Martin Tailless Configuration

However, many control law design engineers are apprehensive of multivariable control methods, for several reasons:

1. Lack of experience in applying multivariable design methods
2. Lack of understanding of relevant theory
3. Limited time and budget to learn the new approaches
4. Ability to get classical control to work, although the resulting control may not be as effective or as easily designed as if multivariable methods had been used

There is an urgent need for a software tool to facilitate integrated control design.

2.3 Technical Approach

The end product of this study is a set of software requirements for an IMPAC design tool. To accomplish this the following steps were taken:

1. Exercise the critical steps in the IMPAC methodology in a design example.

This was accomplished by replicating central parts of a previous NASA–Lewis design effort for an integrated airframe/engine model of the Lockheed Martin Tactical Aircraft Systems E–7D aircraft in

transition. MATRIX$_X$® "EXEC" files were created to document each step of the IMPAC procedures.

2. Document the design and analysis procedures.

Each step of the design and analysis procedures was documented with MATRIX$_X$® output data, plots, and diagrams. For each step, lessons learned were listed along with recommendations that would improve the procedure.

3. Prepare a set of software requirements for a user–friendly tool.

The functional characteristics and interfaces were determined from the documented design and analysis procedures. Prototypes for a graphical user interface (GUI) were diagramed to specify how the tool would interact with the user. Finally, a trade study was performed to assess custom software development versus a shell built around a commercial product (i.e. MATRIX$_X$®, MATLAB®).

2.4 Document Overview

Section 3 of this document is a general summary of the IMPAC design methodology. Section 4 is a step–by–step description of the application of IMPAC to the E–7D STOVL aircraft. This application was an attempt to replicate control design work with this aircraft model at NASA Lewis Research Center. Our conclusions and recommendations concerning each step of this process are included. Section 5 contains extended recommendations for developing the IMPAC design software tool. Overall summary and conclusions are in Section 6. After References, Section 8 is an appendix to Section 4 and lists the MATRIX$_X$® functions we developed to implement IMPAC for the E–7D model.

---

eXceed/NT™ is a trademark of Hummingbird Communications, LTD.
MathScript™ is a trademark of Integrated Systems, Inc.
MATLAB® is a registered trademark of The MathWorks, Inc.
MATRIX$_X$® is a registered trademark of Integrated Systems, Inc.
Motif™ is a trademark of Open Software Foundation.
Pentium® is a registered trademark of the Intel Corporation.
SIMULINK™ is a trademark of The MathWorks, Inc.
SystemBuild™ is a trademark of Integrated Systems, Inc.
VAX™ is a trademark of Digital Equipment Corporation.
VAXstation™ is a trademark of Digital Equipment Corporation.
Windows™ is a trademark of Microsoft Corp.
Xmath™ is a trademark of Integrated Systems, Inc.
X Window System™ is a trademark of Massachusetts Institute of Technology.
Xμ™ is a trademark of Integrated Systems, Inc.
μ–Tools™ is a trademark of Musyn, Inc.

3. IMPAC METHODOLOGY OVERVIEW

3.1 Introduction

The IMPAC methodology, developed by Garg *et al.* [1,2,3], is an approach to integrated airframe and propulsion control design that was developed with the intent of combining the "best" aspects of prior centralized [4,5] and decentralized [6,7] approaches. This approach consists of first designing a centralized controller considering the airframe and propulsions systems as one integrated system, and then partitioning the centralized controller into decentralized subcontrollers with a specified interconnection structure. The centralized design accounts for all the subsystem interactions and serves as a benchmark of performance to judge the partitioned subcontrollers. The centralized controller is partitioned into separate airframe and propulsion subcontrollers for ease of implementation and to allow for independent closed–loop propulsion system validation (Figure 3.1).

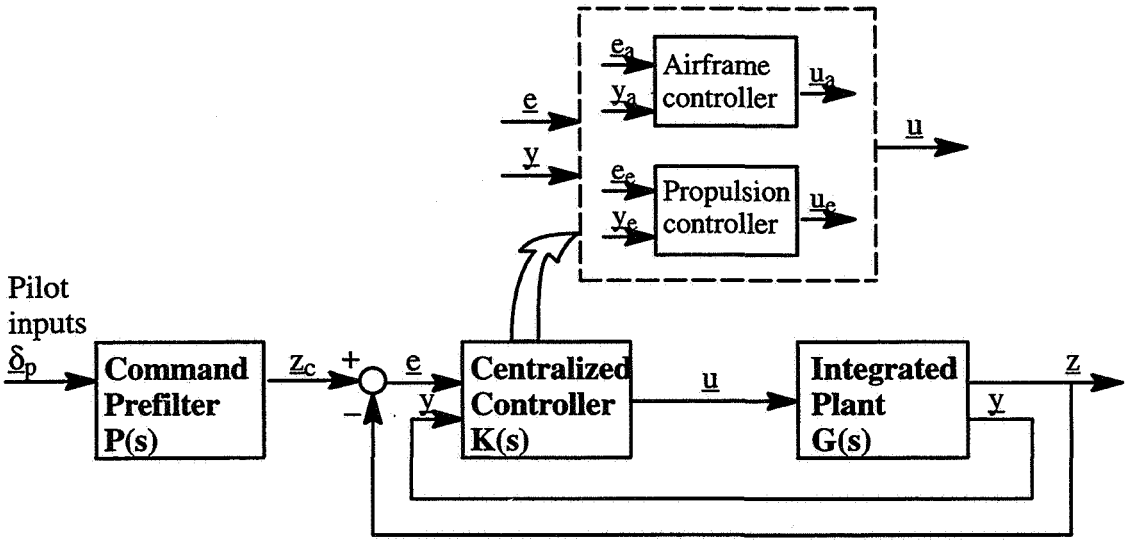


Figure 3.1 IMPAC Methodology

The IMPAC design process involves several discrete steps, as illustrated in Figure 3.2. The following is a brief step–by–step description of the IMPAC design methodology.

3.2 Centralized Controller Design

3.2.1 Full–Order Centralized Controller Design

The centralized controller synthesis is based on the H_∞ design technique. The design problem is a general command tracking and disturbance rejection problem, but it has been previously determined

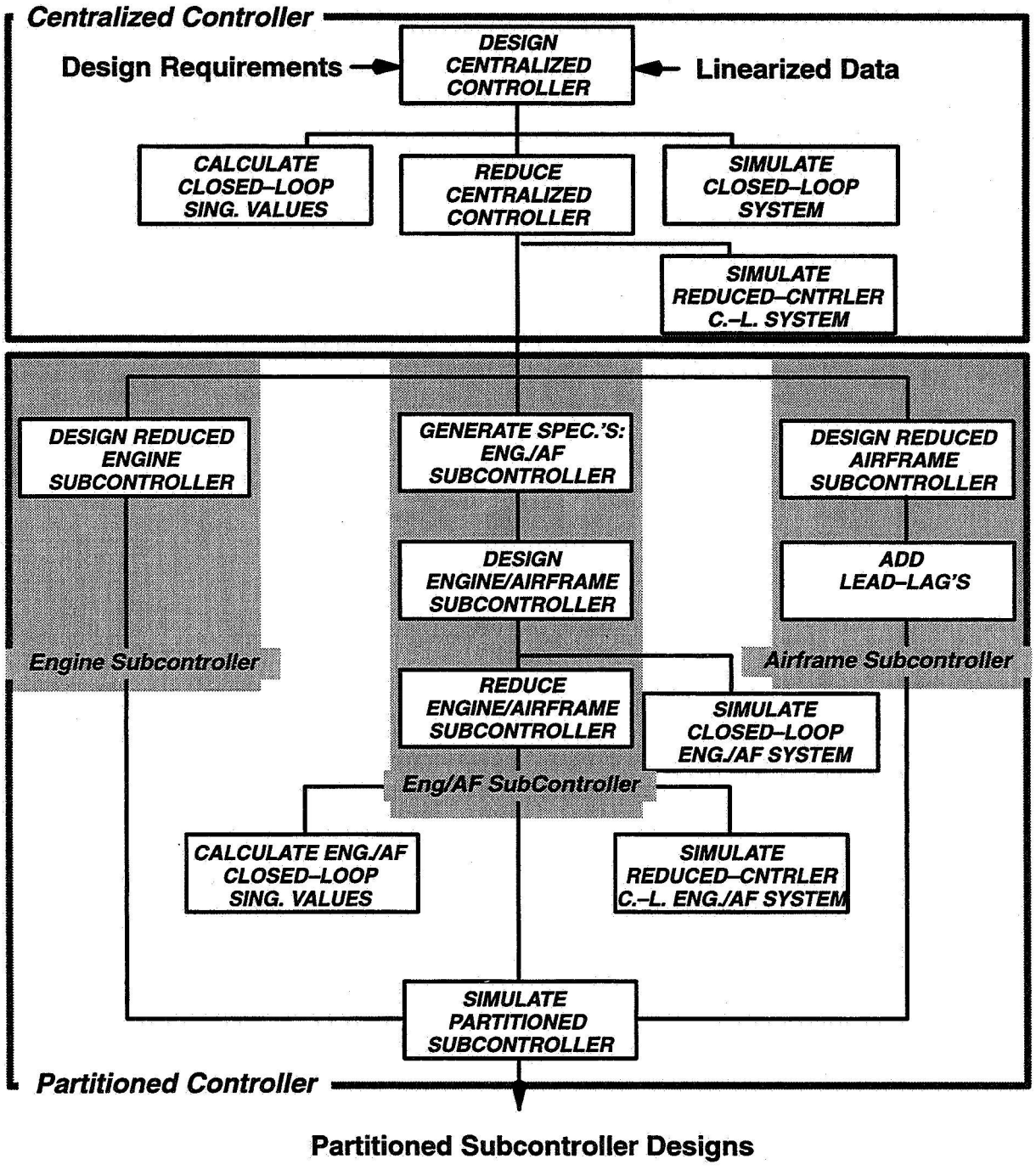


Figure 3.2 IMPAC Design Process

that mixed–sensitivity H_∞ control design is an effective way to accomplish the design. Proper formulation using H_∞ theory provides for building in stability robustness and obtaining an adequate trade–off between performance and allowable control power in the resulting controller.

The three transfer functions that are of interest for such a problem are the sensitivity function, S(s), the complementary sensitivity function, T(s), and the control transmission function, C(s). These represent the closed–loop transfers from the reference commands and additive plant measurement noise to, respectively, tracking errors, controlled variables, and commanded control inputs. In order to influence both the low–frequency and high–frequency properties of the closed–loop system, it is desirable to find a controller K(s) that minimizes a weighted norm of a combination of these three transfer functions, i.e., K(s) represents

$$\min_{\text{Stabilizing } K(s)} \|H(s)\|_\infty$$

where

$$H(s) = \begin{bmatrix} W_S(s) \cdot S(s) \\ W_T(s) \cdot T(s) \\ W_C(s) \cdot C(s) \end{bmatrix}$$

and the infinity norm

$$\|H(s)\|_\infty = \sup\left(\sigma_{max}[H(j\omega)]\right).$$

The weighting functions $W_S(s)$, $W_T(s)$, and $W_C(s)$ are used to tune the controller K(s) such that the design objectives are met. The H_∞ formulation allows for feedback of plant measurements other than just tracking errors as inputs to the controller. Figure 3.3 shows these feedbacks as $\underline{y}$. Figure 3.4 illustrates the closed–loop vehicle and centralized controller system.

The infinity norm of H(s) can be used to evaluate whether the control design objectives have been met. In general, $\|H(s)\|_\infty \leq 1$ indicates that all the design specifications formulated through the various weightings will be met. $\sigma_{max}[W_S(j\omega)\,\underline{e}(j\omega)]$, $\sigma_{max}[W_T(j\omega)\,\underline{z}(j\omega)]$, $\sigma_{max}[W_u(j\omega)\,\underline{u}(j\omega)]$, and $\sigma_{max}[W_{\dot{u}}(j\omega)\,\underline{\dot{u}}(j\omega)]$ with commands $\underline{z}_c$ as inputs have also been used to evaluate the centralized controller [2].

3.2.2 Centralized Controller Reduction

The H_∞–derived centralized controller can generally be reduced in order. Modal residualization and internally balanced realization reduction techniques have been used in the past for this purpose.

Minimum and maximum singular value analysis of the closed–loop tracking system, T(s), where $\underline{z}(s) = T(s)\,\underline{z}_c(s)$, has been used to validate the order reduction. Time–domain simulations have also been useful for this evaluation, with the goal of showing decoupled command tracking bandwidths and reasonable control actuation requirements even when nonlinearities are included in the model. Detailed μ stability robustness evaluation of the control system with respect to variations in the plant system A and B matrices has also been used to evaluate the robustness of the reduced centralized controller [2].

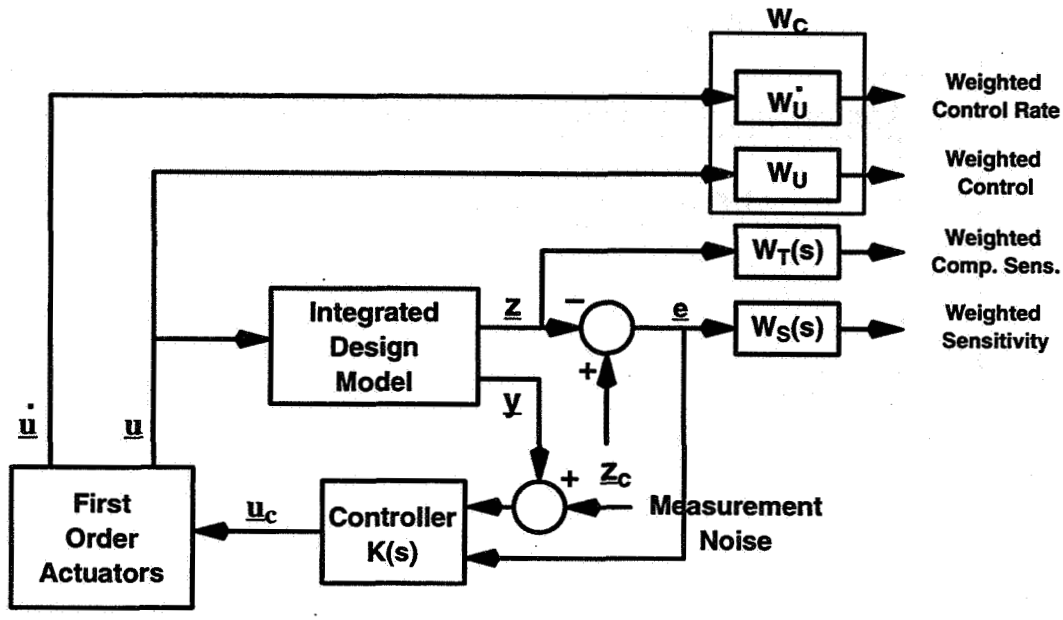


Figure 3.3 H∞ Controller Synthesis Formulation

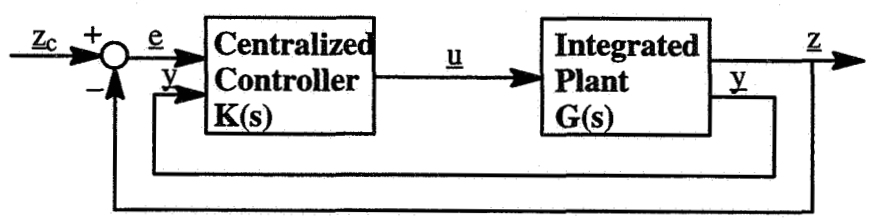


Figure 3.4 IMPAC Centralized Controller/ Plant System

3.3 Subcontroller Partitioning

The controller partitioning task requires that a candidate control structure for the partitioned system be specified. For the integrated flight/ propulsion (IFPC) problem, and as Figure 3.5 illustrates, the assumed control structure is hierarchical, with the airframe control partition issuing commands z_{ea_c} to be tracked through engine control.

3.3.1 Engine Subcontroller Partitioning

The engine subcontroller, $K_e^e(s)$, is obtained as a reduced–order approximation of the $K_{ee}(s)$ block of the centralized controller, when the centralized controller K(s) has been partitioned into four

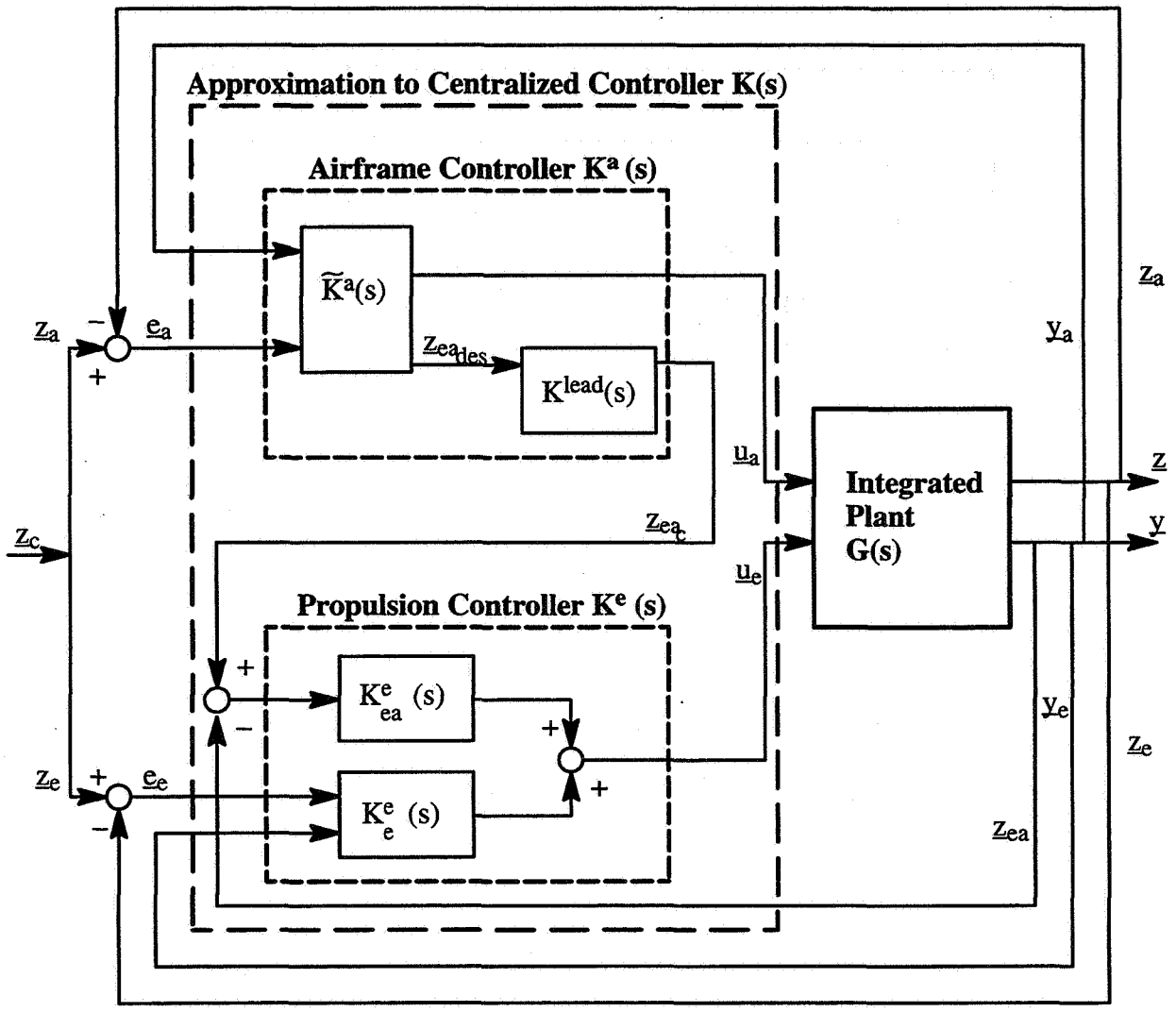


Figure 3.5 IMPAC IFPC Controller Partitioning Structure

blocks:

$$\begin{bmatrix} \underline{u}_a \\ \underline{u}_e \end{bmatrix} = \begin{bmatrix} K_{aa}(s) & K_{ae}(s) \\ K_{ea}(s) & K_{ee}(s) \end{bmatrix} \cdot \begin{bmatrix} \begin{bmatrix} \underline{e}_a(s) \\ \underline{y}_a(s) \end{bmatrix} \\ \begin{bmatrix} \underline{e}_e(s) \\ \underline{y}_e(s) \end{bmatrix} \end{bmatrix}$$

Any suitable order reduction technique, such as the internally balanced realization approach, can be used. To reduce subcontroller complexity, $K^e_e(s)$ should be as low in order as possible while obtaining a good match with the input/output characteristics of $K_{ee}(s)$.

3.3.2 Engine–Airframe Subcontroller Design

The first step in designing the engine–airframe controller is to determine bandwidth requirements on the engine subsystem for tracking the interface variable commands $\underline{z}_{ea_c}$ generated by the partitioned airframe subcontroller. One suggested way for determining these requirements is to study the closed–loop frequency response $T^{ae}_{ea_i}(j\omega)$ from all the airframe commands $\underline{z}_a$ to each individual element $\underline{z}_{ea_i}$ using the centralized controller. A suitable minimum requirement on the tracking bandwidth ω_{ea_i} for the engine subsystem is that $\sigma[T^{ae}_{ea_i}(j\omega)] << 1$ for $\omega > \omega_{ea_i}$. Satisfying this implies that the demand for response in interface variable $\underline{z}_{ea_i}$ required to track the airframe commands $\underline{z}_{a_c}$ will roll off prior to loss in the capability of the engine subcontroller to track the corresponding command $\underline{z}_{ea_{c_i}}$.

In general, there will be other limits on the minimum required tracking bandwidth for the interface variables imposed by requirements such as disturbance rejection, robustness to low–frequency model variations, stability, etc. The maximum achievable tracking bandwidth will normally be limited by control actuation requirements and high–frequency modeling errors.

Another requirement that might suitably be placed on $K^e_{ea}(s)$ is to provide decoupled command tracking of $\underline{z}_{ea}$ without excessive disturbance in $\underline{z}_e$.

The next step is to design $K^e_{ea}(s)$ to meet these specifications. Any control synthesis technique that allows for formulating a mixed command tracking and regulation control problem can be used, although H_∞ techniques have been used by NASA Lewis researchers [8].

3.3.3 Airframe Subcontroller Partitioning

$\tilde{K}^a(s)$ is obtained as an approximation to transfer function for the $[\underline{e}_a\ \underline{y}_a]^T \rightarrow [\underline{u}_a\ \underline{z}_{ea}]^T$ system. This system is suggested in Figure 3.6. The engine subsystem loop should be closed when determining

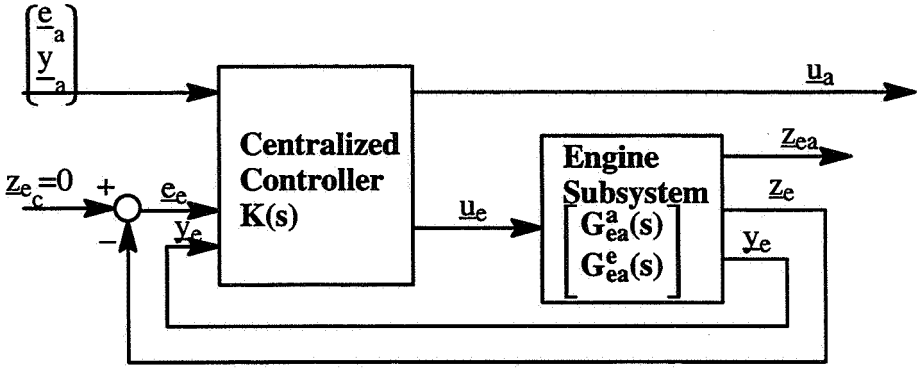


Figure 3.6 Control Loop to Determine $\tilde{K}^a(s)$ Block of Partitioned Airframe Subcontroller [3]

the transfer function. An expression for the overall transfer function can be obtained using algebraic

manipulation of the various subblocks of the centralized controller (see Section 2.2.1 above) and the engine subsystem of the partitioned controlled plant in the open–loop system below:

$$\begin{bmatrix} \begin{bmatrix} \underline{z}_a(s) \\ \underline{y}_a(s) \end{bmatrix} \\ \begin{bmatrix} \underline{z}_e(s) \\ \underline{y}_e(s) \end{bmatrix} \\ \underline{z}_{ea}(s) \end{bmatrix} = \begin{bmatrix} G_{aa}(s) & G_{ae}(s) \\ G_{ea}(s) & G_{ee}(s) \\ G^a_{ea}(s) & G^e_{ea}(s) \end{bmatrix} \cdot \begin{bmatrix} \underline{u}_a(s) \\ \underline{u}_e(s) \end{bmatrix}$$

3.3.4 Lead Compensation

The next step in the IMPAC design process is to add lead filtering to compensate for the limited $\underline{z}_{ea_c}$ tracking bandwidth of the engine subsystem. $\tilde{K}^a(s)$ generates $\underline{z}_{ea_{des}}$, the desired response in the interface variables to airframe controlled variable commands. If $\underline{z}_{ea_{des}}$ were used directly as commands for the interface variables, then the actual $\underline{z}_{ea}$ response with the partitioned subcontrollers would lag the desired response $\underline{z}_{ea_{des}}$, due to the limited tracking bandwidth of the engine subsystem. In general, there will be a trade–off between the amount of lead compensation in $K^{lead}(s)$ and the $\underline{z}_{ea_c}$ tracking bandwidth of the engine subsystem. High lead compensation is undesirable, because it can result in saturation of the engine actuators due to command magnification, whereas low–lead compensation will require large $\underline{z}_{ea_c}$ tracking bandwidth. Since the $K^e_{ea}(s)$ portion of the engine controller provides decoupled tracking of $\underline{z}_{ea_c}$, $K^{lead}(s)$ can simply be of the form below, with a_i and b_i chosen based on the amount of lead desired in $\underline{z}_{ea_i}$.

$$K^{lead}(s) = \text{diag}\left[\frac{s + a_i}{a_i}\ \frac{b_i}{s + b_i}\right],\ a_i < b_i$$

3.3.5 Evaluation of Partitioned Subcontrollers

Closed–loop performance and robustness comparisons between the centralized and partitioned linear controllers are made to validate the partitioning results as well as acceptability of the chosen decentralized control structure.

3.4 Completing Controller Design

Final steps in the IMPAC process would be design and scheduling of the linear partitioned subcontrollers over the operational flight envelope, nonlinear design such as incorporation of limit logic for operational safety, and evaluations of the final full–envelope control design.

4. IMPAC DESIGN EXAMPLE: E–7D STOVL AIRCRAFT

4.1 Introduction

In order to gain familiarity with the IMPAC process and provide relevant feedback and recommendations, we exercised critical steps in the methodology by replicating the first stages of a previous NASA–Lewis design effort for an integrated airframe/engine model of the E–7D aircraft in transition. We generated and reduced a point design for the centralized controller, partitioned it into engine, engine–airframe, and airframe subcontrollers, and reduced the order of these subcontrollers. MATRIX$_X$® "EXEC" files were created to implement each step of the IMPAC design process.

4.2 E–7D STOVL Aircraft

4.2.1 Aircraft Modeling for IMPAC Design

The E–7D STOVL aircraft is illustrated in Figure 4.1. This aircraft has a highly coupled propulsion

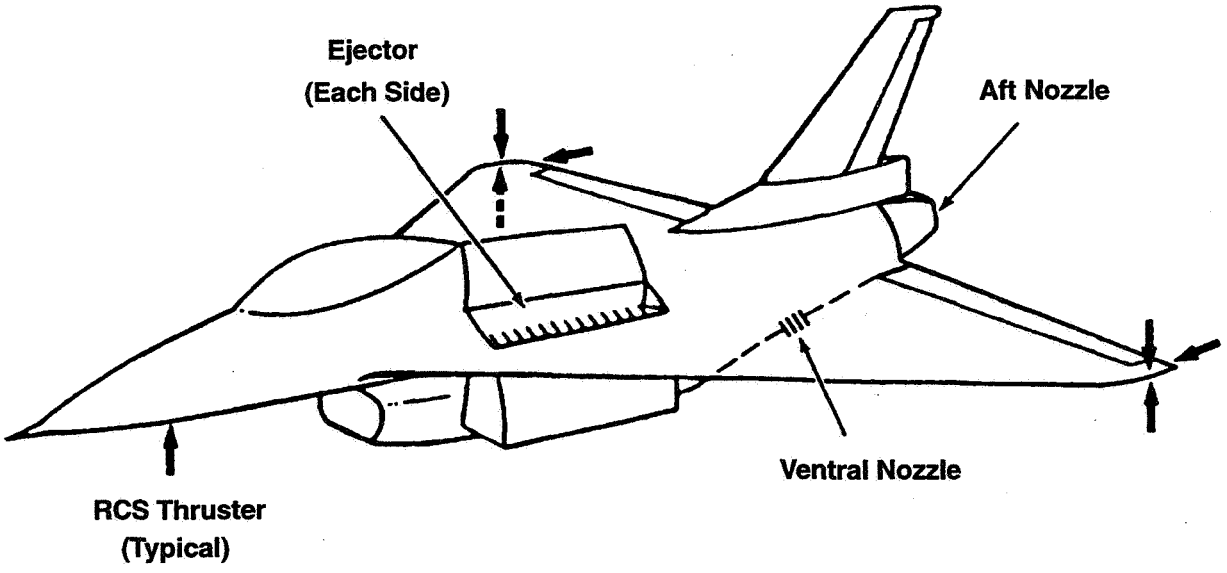


Figure 4.1 E–7D Aircraft Model

system, and it is thus a highly suitable application for the IMPAC design process.

The E–7D aircraft propulsion consists of (1) a two–dimensional convergent–divergent vectoring aft nozzle with afterburner for conventional flight; (2) ejectors powered by mixed engine flow for

propulsive lift during transition and hover; (3) a vectoring ventral nozzle for pitch control and lift augmentation during transition and hover; and (4) a roll/ pitch/ yaw jet reaction control system (RCS) powered by engine compressor bleed flow for attitude control during hover.

Only the longitudinal dynamics were considered in this contract. The 11–element longitudinal state vector is ([3])

$\underline{x}$ = [N2, N25, Tmhpc, Tmpc, Tmhpt, Tmlpt, u, w, q, θ, h]^T

where

N2 = Engine Fan Speed, rpm
N25 = High Pressure Compressor Speed, rpm
Tmhpc = High Pressure Compressor Metal Temp., °R
Tmpc = Burner Metal Temp., °R
Tmhpt = High Pressure Turbine Metal Temp., °R
Tmlpt = Low Pressure Turbine Metal Temp., °R
u = Axial Velocity, ft/s
w = Vertical Velocity, ft/s
q = Pitch Rate, rad/s
θ = Pitch Attitude, rad
h = Altitude, ft

The control inputs, partitioned in the NASA Lewis work into four airframe and four engine control inputs, are

$\underline{u}_a$ = [δe, AQR, ANG79, ANG8]^T
$\underline{u}_e$ = [WF, A8, ETA, A78]^T

where

δe = Elevator Deflection, deg
AQR = Pitch RCS Area, in^2
ANG79 = Ventral Nozzle Vectoring Angle, deg
ANG8 = Aft Nozzle Vectoring Angle, deg
WF = Fuel Flow Rate, lb_m/hr
A8 = Aft Nozzle Area, in^2
ETA = Ejector Butterfly Valve Angle, deg
A78 = Ventral Nozzle Area, in^2

The controlled outputs for the airframe and engine systems were chosen to be

$\underline{z}_a$ = [Vv, Qv, γ]^T
$\underline{z}_e$ = [N2]

where

$$Vv = \dot{V} + 0.1\,V$$
$$Qv = q + 0.3\,\theta$$

where

$\dot{V}$ = Acceleration Along Flight Path, ft/s^2
V = True Airspeed, ft/s
γ = Flight Path Angle, deg

This blending of controlled variables was chosen to provide the response types that are desirable for good handling qualities.

The 10 total inputs to the airframe and engine controllers are the tracking errors $\underline{e}_a$ (3–vector) and $\underline{e}_e$ (1–vector) corresponding to $\underline{z}_a$ and $\underline{z}_e$, and the measurement feedbacks

$$\underline{y}_a = [V, \dot{V}, \theta, q, \gamma]^T$$
$$\underline{y}_e = [N2, WB3]^T$$

where WB3 is the engine bleed bleed flow demand from the RCS.

This set of measurement feedbacks includes more feedbacks than would be used for traditional airframe–only aircraft control augmentation but may include feedbacks that would be necessary for a STOVL configuration.

The interface from the propulsion system model to the airframe model was chosen to be the gross thrusts from three of the four engine systems (excluding the RCS), i.e.,

$$\underline{z}_{ea} = [FG9, FGE, FGV]^T$$

where

FG9 = Aft Nozzle Gross Thrust, lb_f
FGE = Ejector Gross Thrust, lb_f
FGV = Ventral Nozzle Gross Thrust, lb_f

The design point used corresponds to an 80 kt condition during a decelerating transition to hover on landing approach. The linear model matrices for this design point were taken from reference [2]. To ensure that maximum allowable or safe control levels would not be exceeded, the design plant inputs were normalized when needed by the inverses of the maximum allowable deflections, $\underline{u}_{max}$. Normalizing values for the controlled outputs, $\underline{z}_{c_{max}}$, were chosen to ensure that each element of $\underline{z}$

could be commanded individually to its maximum value within its frequency range of interest without any of the control inputs exceeding $\underline{u}_{max}$. The normalizing terms were given in [2].

An unmodified linear design model would not account for the absolute nonlinearity from RCS area commands to engine bleed flow demand, WB3, if all (three–axis) RCS thrusters were being modeled. The modified model used here has zeros substituted for the original elements of the control effectiveness matrix, B, from the pitch RCS area to engine states. The engine bleed flow demand associated with pitch RCS area commands was then modeled as a first–order–filtered external disturbance affecting the engine dynamics through every engine state. For H_∞ control synthesis, the exogenous input was scaled when needed by the maximum possible RCS bleed flow. The bleed flow was used as a feedback to the controller, since it was assumed that an analytic approximation would be available given a known RCS command.

Actuator characteristics were taken from Ref. [2]. These models were first–order approximations to rate–limited full–order actuator models derived from describing function analysis. Figure 4.2

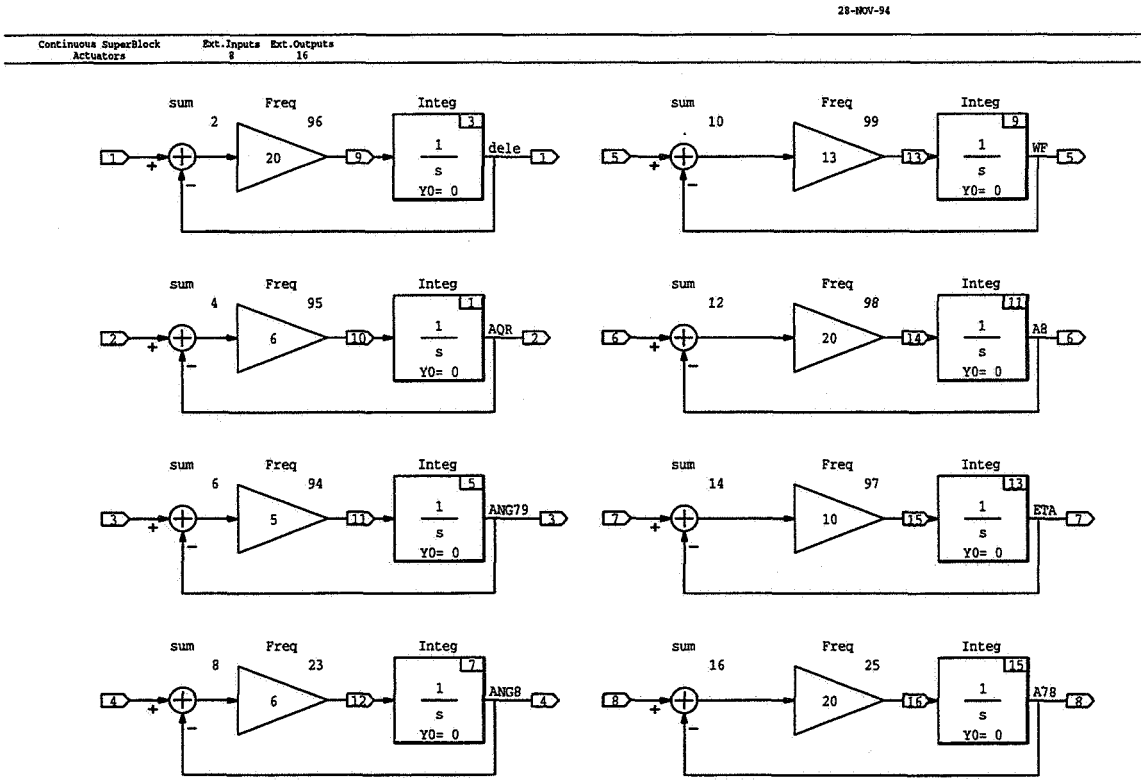


Figure 4.2 Actuator Models

illustrates the MATRIX$_X$® actuator models for the H_∞ centralized controller design, with airframe actuators on the left side of the figure, engine actuators on the right side.

The design plant without sensitivity and complementary sensitivity weightings is 20th–order: 11th–order integrated longitudinal airframe/ propulsion model, first–order actuators for the 8 control inputs, and the first–order filter for the bleed flow disturbance.

4.2.2 Conclusions and Recommendations Regarding Modeling for IMPAC Design

Knowing the control partitioning and hierarchy as well as the command variables and feedbacks greatly simplified replicating the NASA Lewis work. (Much of our control design work at LMTAS can involve defining command variables and defining and tailoring control modes, and developing mode switch logic, especially with STOVL configurations.) Determining a suitable partitioning for the control vector and choosing $\underline{z}_c$ and partitioning it into $\underline{z}_a$ and $\underline{z}_e$ would have been a trial and error process otherwise, as would selecting $\underline{y}$. There are roughly as many command variables, $\underline{z}$, as control inputs, $\underline{u}$. Several types of analysis could be used to help partition a system, e.g., modal analysis, relative gain array analysis, Grammian analysis, and singular value analysis. Scaled open–loop airframe singular values are shown in Figure 4.3, and scaled open–loop engine singular values are

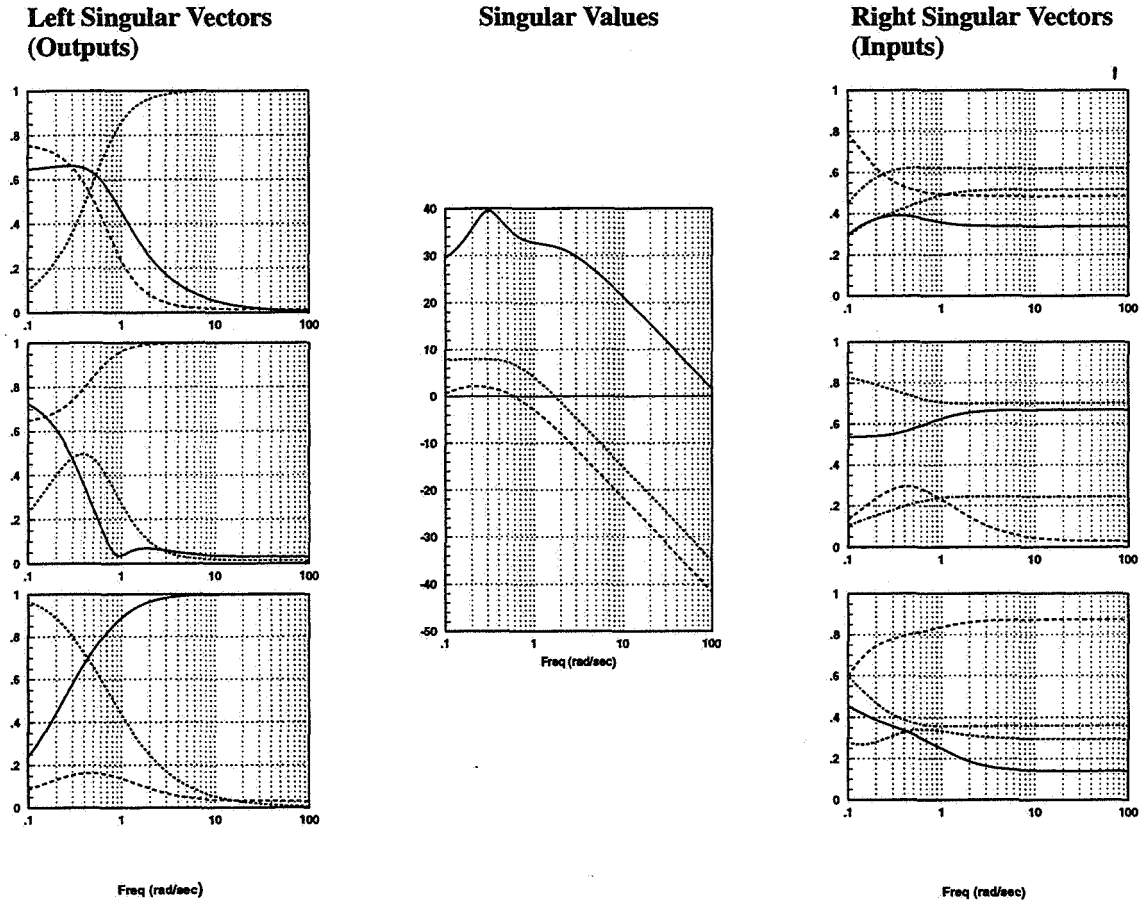


Figure 4.3 Scaled Open–Loop Airframe Singular Values

shown in Figure 4.4. Open–loop singular value analysis can also be used to help set control design

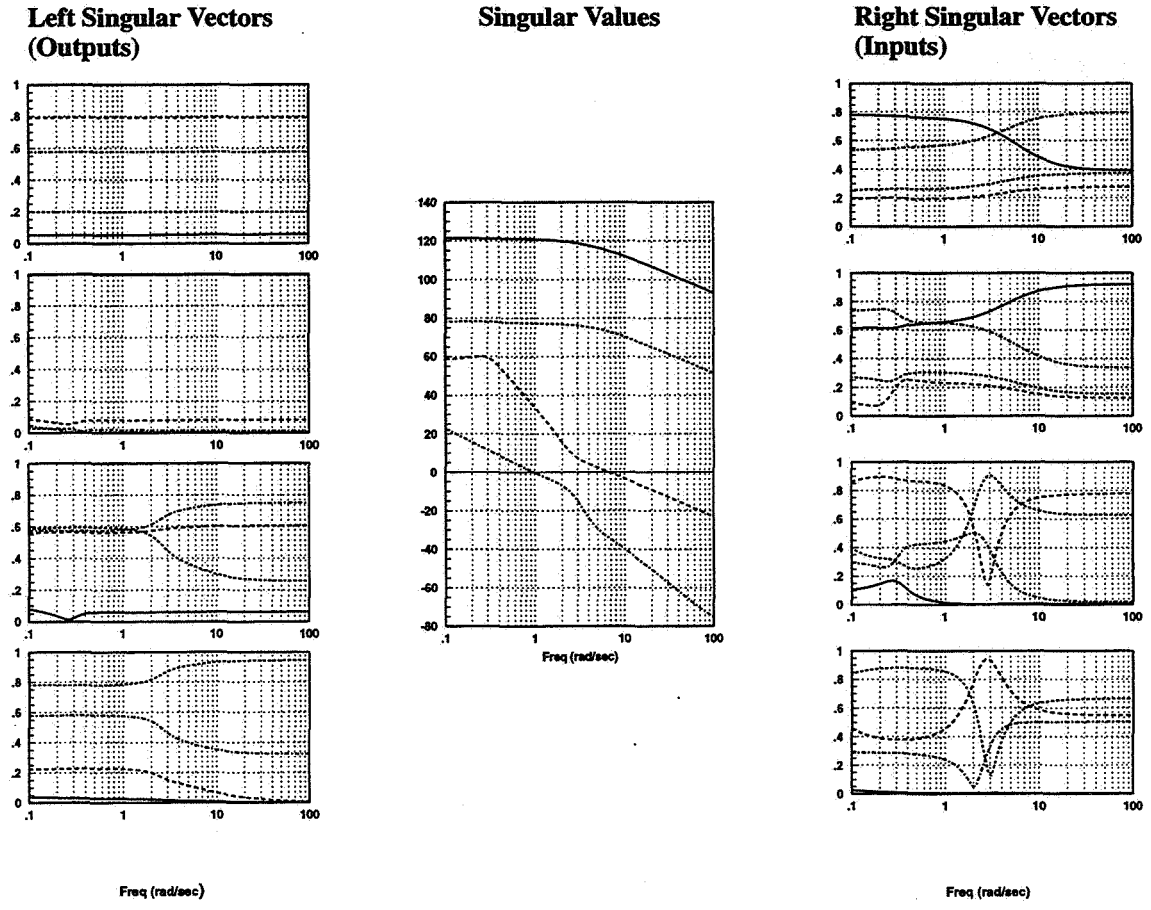


Figure 4.4 Scaled Open–Loop Engine Singular Values

specifications for the various subsystems.

The choice of measurement feedbacks selected by NASA Lewis researchers are such that $\underline{y}_a$ and $\underline{y}_e$ are basically in one–to–one relationship with $\underline{z}_a$ and $\underline{z}_e$, respectively. Making available procedures for systematically determining controller partitioning and hierarchy would be highly recommended.

In the last decade, we have consistently designed controllers to command a few generalized controls, e.g., separate controls to effect accelerations in translational and rotational degrees of freedom, and incorporated a control selector after the regulator to distribute the generalized commanded forces and moments to the actual available control effectors. Our usual control law architecture suggests

this separation of regulation from control selecting (Figure 4.5). This approach has several

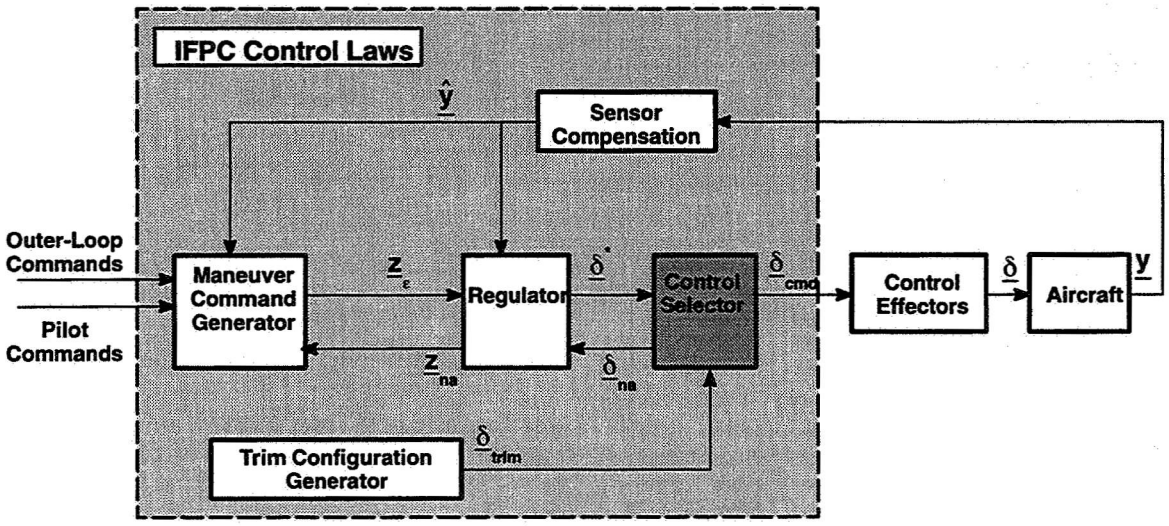


Figure 4.5 Lockheed Martin Tactical Aircraft Systems Control Law Structure

advantages:

1. The regulator is smaller, thus generally easier to design.
2. The regulator is smaller, so scheduling across different flight conditions means that fewer regulator gains must be interpolated. The scheduled gains also seem to change in a more regular fashion than in regulators designed directly for the actual control effectors.
3. Reconfiguring controls after a control failure is more straightforward.
4. Commands can be reassigned when actuators saturate.
5. Unmet commands can be computed and used to reduce integrator windup.

We recommend investigating designing the IMPAC control laws using generalized controls. This will involve more than a trivial change to IMPAC. Reference [9] shows how generalized controls were chosen for our control design work with the same E–7D aircraft and develops the control selector.

4.3 Overview of IMPAC Process for E–7D

The IMPAC design process involves several discrete steps, as illustrated in Figure 4.6. MATRIX$_X$® "EXEC" files were written to implement each step of the overall design process. The files associated with each step are also indicated in this figure.

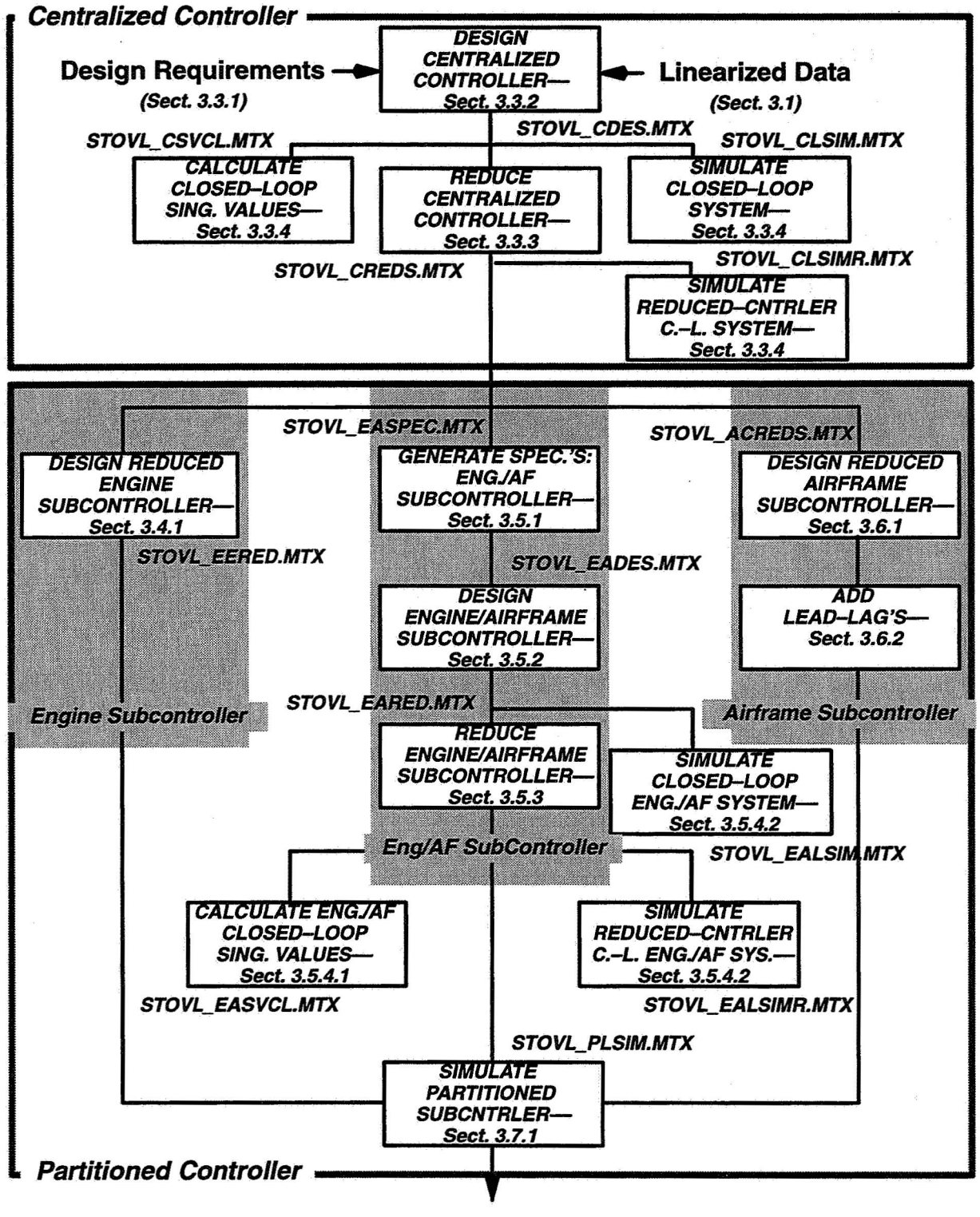


Figure 4.6 IMPAC Design Process and MATRIX$_X$® "EXEC" Files for E–7D Application

4.4 Centralized Controller Design

4.4.1 Augmenting H∞ Design Plant

Measurement noises were added to satisfy a necessary condition (rank (D_{21}) = number of feedbacks to the controller) for the two Riccati equation state–space solution to the H_∞ control design problem. Very small noise magnitude (.001) was chosen for the measurement noises to have negligible effect on the resulting controller.

Sensitivity and complementary sensitivity weightings (W_S and W_T) used in the centralized controller design process were taken from Ref. [2]. The sensitivity and complementary sensitivity weights were chosen to be first–order transfer functions, in order to provide adequate frequency response shaping without overly increasing the resulting controller order.

For each controlled variable, the W_S zero and pole were chosen to result in a low–frequency gain of 1000, gain crossover frequency of 3–4 times the control bandwidth desired for good handling qualities, and a high–frequency gain of 0.1. This choice reflects the desire to synthesize a sensitivity function that gives good steady–state tracking in the presence of disturbances and low–frequency modeling errors, good tracking up to the desired control bandwidth, and reduced emphasis on tracking at high frequencies, where there are significant modeling errors and uncertainties. Figure 4.7 shows the sensitivity weightings.

The W_T weightings were chosen to obtain a low–frequency gain of 0, gain crossover frequency of approximately 1.2 times the corresponding W_S gain crossover frequency, and a high–frequency gain of 1000. This choice ensures that the plant command variable outputs are not penalized at low frequencies, where command tracking is to be emphasized, while at high frequencies these outputs are penalized heavily to provide controller gain attenuation for robustness to high–frequency unmodeled dynamics. Figure 4.8 shows the complementary sensitivity weightings.

As discussed in [2], the control and control rate weightings, W_u and $W_{\dot{u}}$, were chosen to be diagonal matrices containing inverses of elements of $\underline{u}_{max}$ and $\underline{\dot{u}}_{max}$, respectively (see [2] for $\underline{u}_{max}$ and $\underline{\dot{u}}_{max}$ values). Weighting the control rates not only prevents synthesizing controllers with high rate requirements but also ensures satisfaction of a necessary condition (rank(D_{12}) = number of control inputs $\underline{u}$) for solving the H_∞ control problem using the two Riccati equation state–space solution algorithm.

The design plant was now 28th–order: 20th–order plant plus first–order sensitivity and complementary sensitivity weights for the 4 controlled variables. The MATRIX$_X$® design plant model for H_∞ centralized controller design is shown in Figure 4.9.

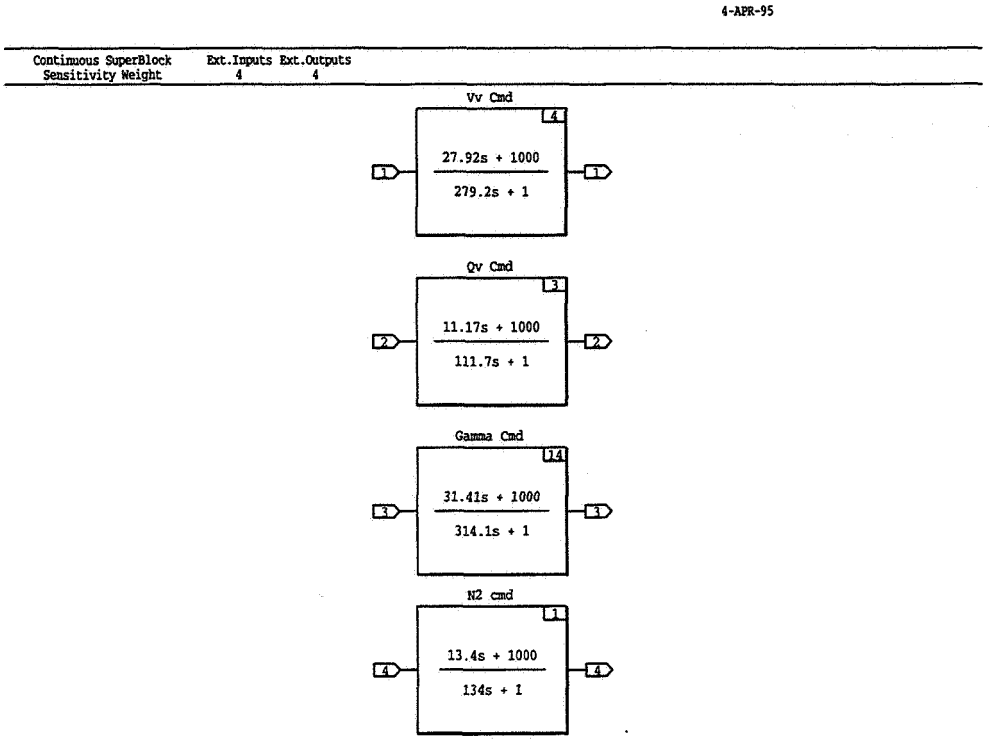


Figure 4.7 Sensitivity Weighting Function Models

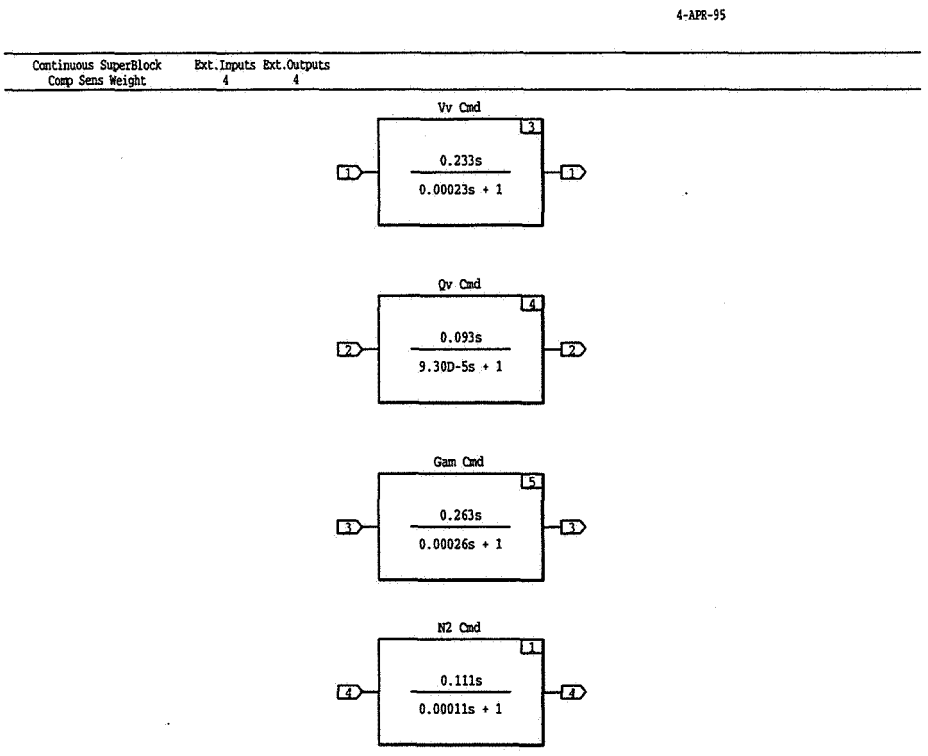


Figure 4.8 Complementary Sensitivity Weighting Function Models

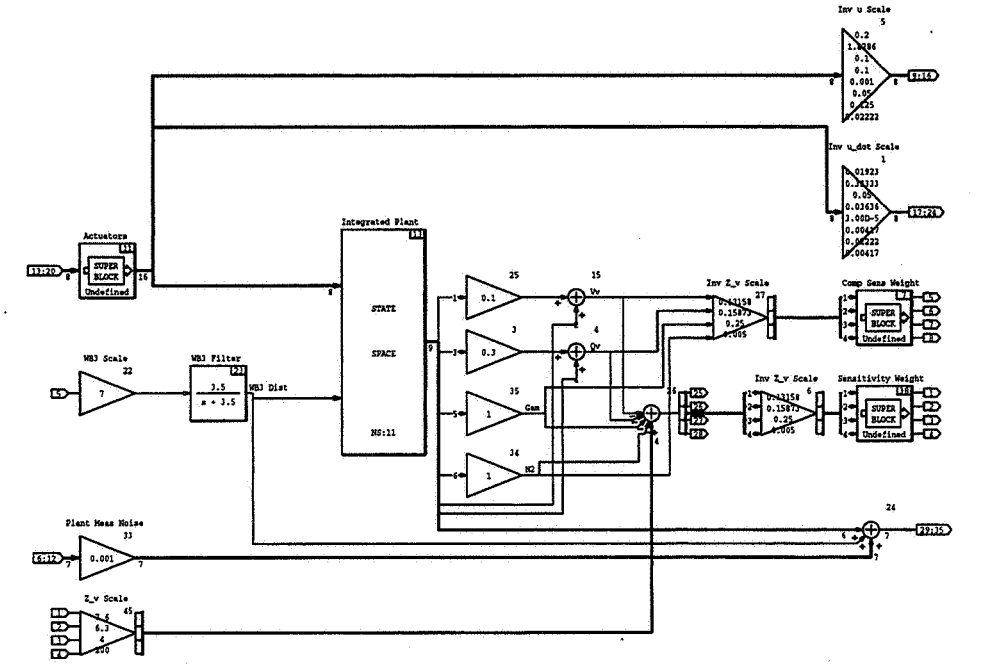


Figure 4.9 Longitudinal Design Plant

The design plant for the standard H_∞ control design problem is given by

$$\dot{x} = A\,x + B_1\,w + B_2\,u$$
$$z = C_1\,x + D_{11}\,w + D_{12}\,u$$
$$y = C_2\,x + D_{21}\,w + D_{22}\,u$$

where

x is the states (28)

w is all 12 commands and disturbances—tracked variables $\underline{z}_{cmd}$ (4) as well as WB3 (1) and measurement noise (7)

u is control inputs (the commands to the actuators) (8)

z is all 24 outputs—weighted errors $\underline{e}$ (4), weighted tracked outputs $\underline{z}$ (4), and weighted controls $\underline{u}$ (8) and control rates $\dot{\underline{u}}$ (8)

y is all 11 feedbacks—weighted errors $\underline{e}$ (4) and measurement outputs $\underline{y}$ (7)

This model is referenced directly in the MATRIX$_X$® "EXEC" files in the Appendix.

With a 28th–order design plant, the H_∞ centralized controller obtained using the algorithm of Doyle *et al.* (referenced in [2]) would be 28th–order.

4.4.2 H∞ Centralized Controller Synthesis

The main MATRIX$_X$® "EXEC" file for centralized H_∞ controller solution is listed in Appendix 8.1.1. The computations use modified versions of the HINF_CONTR, SINGRICCATI, SPLIT9, SPLIT4, and CLSYS functions of the Robust Control Module software developed by Integrated Systems, Inc. (referenced in [2]). The functions were modified by NASA Lewis to improve numerical convergence and are available at no cost from NASA Lewis Research Center for MATRIX$_X$® users.

To use the centralized controller design process, the user must specify successively lower input values for the bound on H_∞ until the controller can no longer be computed. The input bound value can then be increased slightly to get a close–to–minimum–norm solution.

4.4.3 Centralized Controller Reduction

The closed–loop MATRIX$_X$® model for the full–order centralized controller is shown in Figure 4.10.

The MATRIX$_X$® "EXEC" file for centralized controller reduction is listed in Appendix 8.1.2. The NASA Lewis order–reduction process was replicated exactly. As this file shows, the input

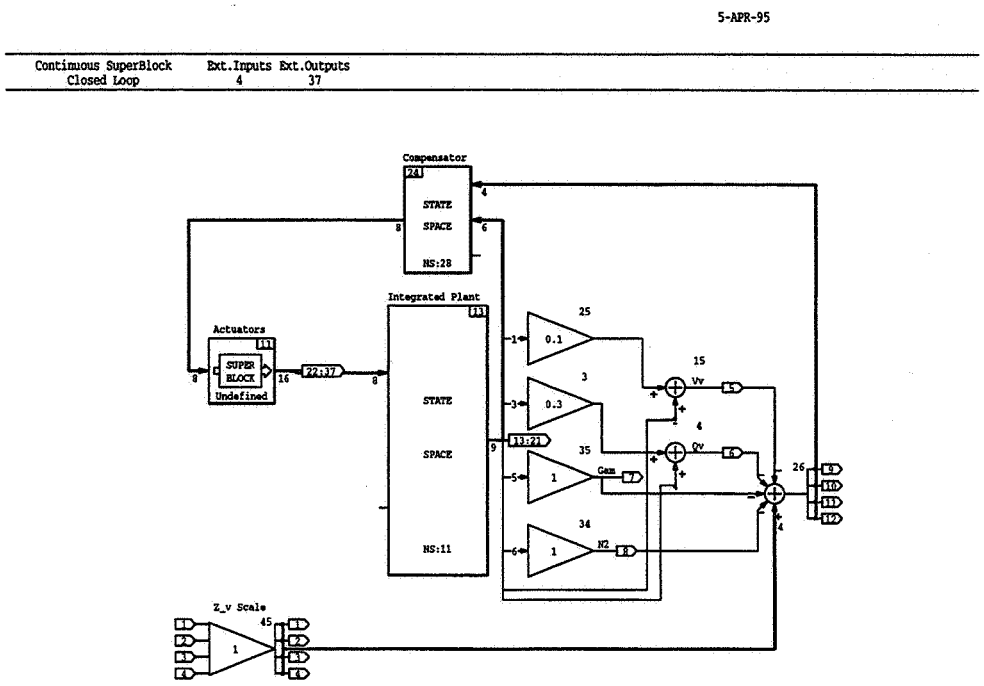


Figure 4.10 Centralized Controller Closed–Loop Model

centralized controller was converted to a modal form, then MREDUCE'd to order 22 (the controller transfer function for the order 20 and 24 solutions did not differ significantly from the order 22 solution), BALANCE'd, and MREDUCE'd again. The first modal residualization was done to truncate higher–order modes before internal balancing. Internal balancing without this first step could not give a solution. Singular values of the original and reduced controllers were plotted, and the minimum and maximum singular values of the original and reduced–order centralized controllers were overplotted for comparison. The singular values of the closed–loop longitudinal system with the original and reduced–order controllers were plotted separately, and the minimum and maximum singular values were then plotted together for direct comparison.

4.4.4 Closed–Loop Analysis of Centralized Controller

The MATRIX$_X$® "EXEC" file for closed–loop linear simulation of the closed–loop longitudinal system plus centralized controller is listed in Appendix 8.1.3.1. Using this file, the user can plot a choice of linear responses to unit step augmented velocity Vv, augmented pitch rate Qv, flight path angle, or engine fan speed commands. Responses of command variables, control inputs, and measurement feedbacks were all plotted. The similar MATRIX$_X$® "EXEC" file for closed–loop linear simulation of the closed–loop longitudinal system plus reduced–order centralized controller is listed in Appendix 8.1.3.2.

The MATRIX$_X$® "EXEC" file for determining closed–loop singular value of the longitudinal system plus full–order centralized controller is listed in Appendix 8.1.3.3. This file provides for

computing several sets of singular values: all inputs to all outputs, $\underline{z}_c$ command inputs to $\underline{z}_c$–$\underline{z}$ error outputs, $\underline{z}_c$ inputs to $\underline{z}$ outputs, and $\underline{z}_c$ inputs to $\underline{u}$ and $\dot{\underline{u}}$ outputs. The maximum singular values for all these cases were then plotted together for comparison.

4.4.5 Conclusions and Recommendations Regarding Centralized Controller Design

Centralized controller design using the IMPAC process was straightforward given W_S, W_T, and the vehicle model. Choosing the W_S and W_T weightings for a new problem might require some experimentation and experience with H_∞ design generally.

We evaluated a 10th–order reduced centralized controller (NASA Lewis reduced the controller to this order) and felt that the frequency and step tracking response were not as good as we would like. We chose a more conservative approach, reducing the centralized controller to order 16 instead of order 10. Figure 4.11 and Figure 4.12 illustrate that controller and closed–loop tracking singular values for the original and 16th–order reduced controller are very similar. This 16th–order controller was partitioned in the next phase of the design work.

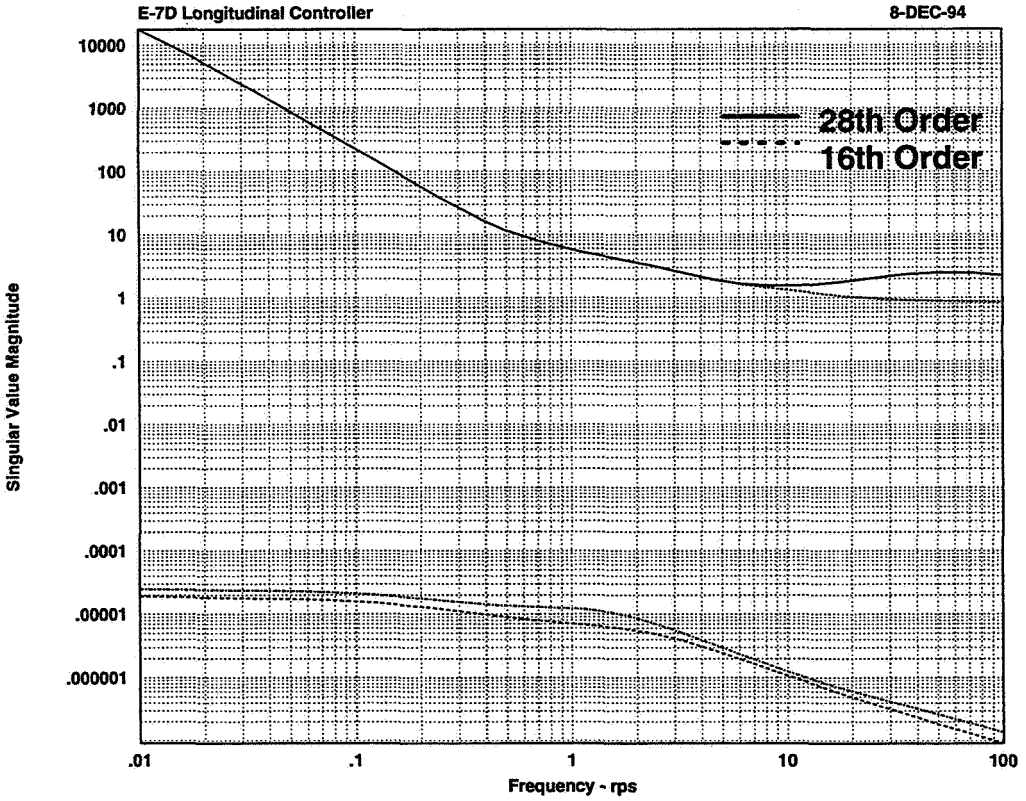


Figure 4.11 Minimum and Maximum Singular Values: Original and 16th–Order Reduced Centralized Controllers

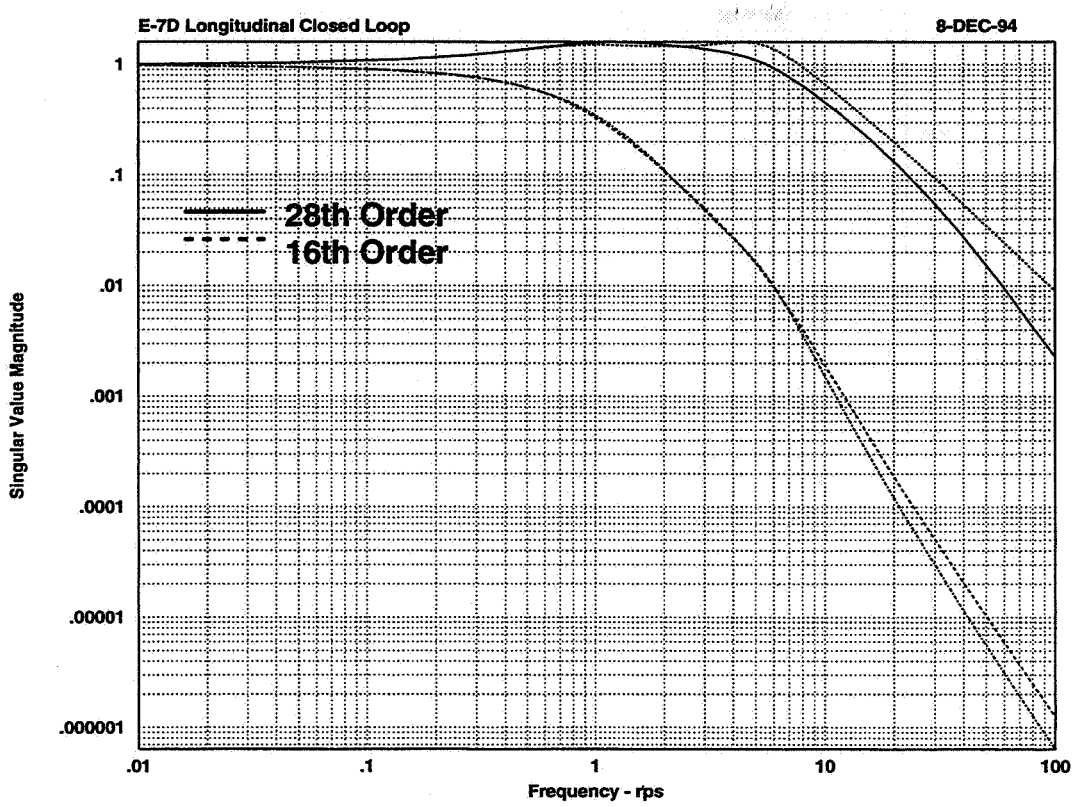


Figure 4.12 Minimum and Maximum Singular Values: Closed–Loop System with Original and 16th–order Reduced Centralized Controllers

4.5 Engine Subcontroller Design

4.5.1 Engine Subcontroller Partitioning

The Matrix$_X$ "EXEC" file for engine subcontroller partitioning is listed in Appendix 8.2.1. Here, the engine partition, i.e., compensation from $\underline{e}_e$ and $\underline{y}_e$ to $\underline{u}_e$, was isolated from the centralized controller, and its singular values were computed and plotted. This partition was then scaled by the inverse of the maximum values of the inputs to the controller, i.e., inv(diag[$N2_{max}$ $N2_{max}$ $WB3_{max}$]), and, on the output side, scaled by the maximum values of the controller outputs $\underline{u}_e$, i.e., diag[WF_{max}, $A8_{max}$, ETA_{max}, $A78_{max}$]. The controller was then transformed to an internally balanced realization and truncated to fourth order. The singular values of this reduced–order engine subcontroller were plotted, and its minimum and maximum singular values were compared in a single plot with those of the original full–order engine partition of the reduced–order centralized controller. Figure 4.13 shows the minimum and maximum singular values of the original and reduced–order engine subcontrollers.

4.5.2 Conclusions and Recommendations Regarding Engine Subcontroller Design

The engine subcontroller was the easiest of the partitioned subcontrollers to develop. Based on comparing singular values, we concurred with NASA Lewis researchers that a 4th–order engine subcontroller seemed adequate.

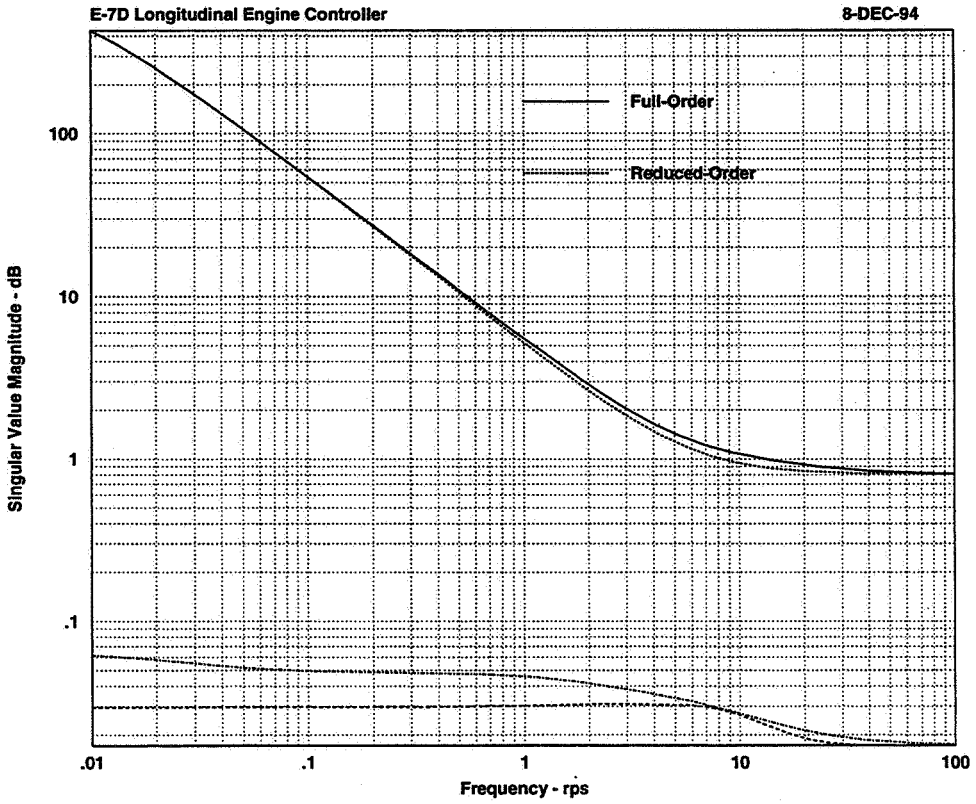


Figure 4.13 Minimum and Maximum Singular Values: Original and Reduced Engine Subcontrollers

4.6 Engine–Airframe Subcontroller Design

4.6.1 Engine–Airframe Subcontroller Design Specification

Engine–airframe interface subcontroller design specifications must be set. The singular values of the frequency responses from all the airframe commands $\underline{z}_a$ to each of the gross thrust engine–airframe interface variables, FG9, FGE, and FGV, were computed, with the engine control loop closed. The MATRIX$_X$® "EXEC" file for developing these singular values is listed in Appendix 8.2.2.1. The three relevant single–output transfer functions were extracted and scaled in this file. The singular values were plotted on a single plot.

As Figure 4.14 shows, the demand for all these gross thrusts rolls off near 1 rad/s and is well below –3 dB (standard bandwidth criterion) for frequencies above 4.5 rad/s. Thus, for the design of $K^e_{ea}(s)$, a tracking bandwidth of 4.5 rad/s for each of the three gross thrusts was judged to be adequate to avoid any significant deterioration in airframe command tracking with the partitioned subcontrollers. This bandwidth specification should also be adequate for rejection of the disturbance due to RCS bleed flow from the engine and should provide robustness to variations in engine dynamics over the transition flight envelope as well as to high–frequency modeling uncertainties.

The sensitivity and complementary sensitivity weights for each of the three thrusts were chosen to reflect the 4.5 rad/s bandwidth and robustness requirements. First–order weights were chosen to

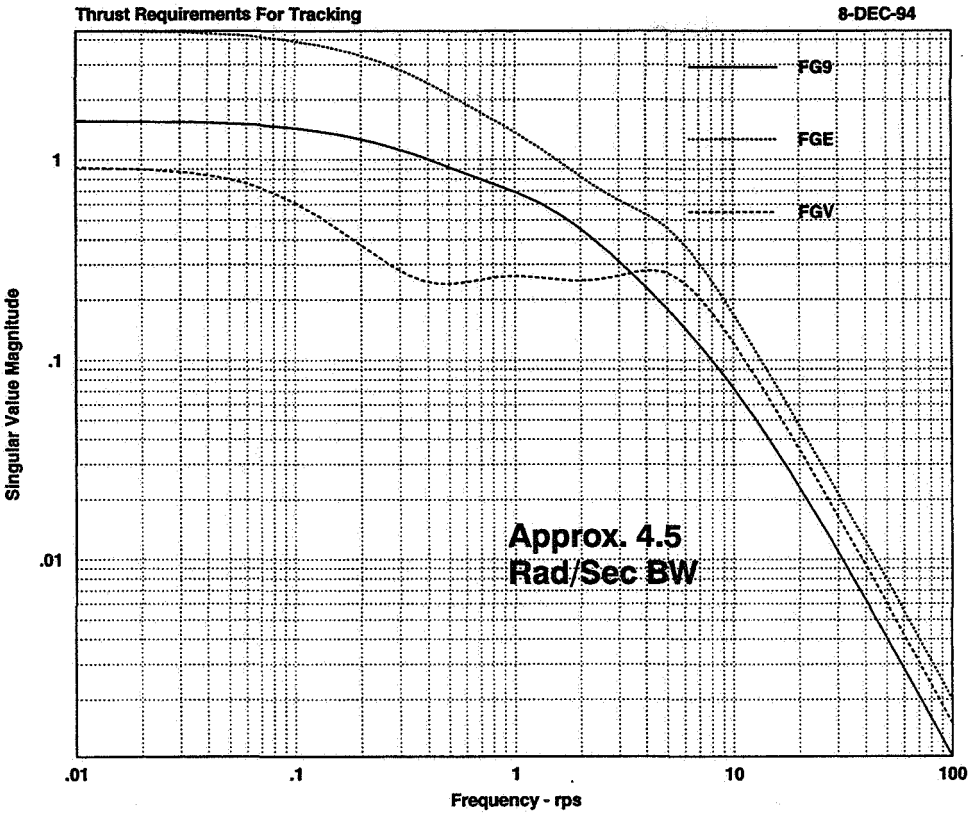


Figure 4.14 Thrust Requirements for Tracking Airframe Commands

simplify the control synthesis. The sensitivity and complementary sensitivity weightings on all three gross thrusts could be chosen to be identical. Sensitivity weights are shown in Figure 4.15 and complementary sensitivity weights in Figure 4.16. The MATRIX$_X$® model for the engine–airframe subcontroller design plant is shown in Figure 4.17.

4.6.2 Engine–Airframe Subcontroller Design

The engine–airframe interface subcontroller, $K^e_{ea}(s)$, was designed using a mixed–sensitivity H_∞ formulation. The controller was intended to provide decoupled tracking of the three thrust commands and regulation of engine fan speed. The first–order engine actuator models (right–hand side of Figure 4.2) were also included in the H_∞ control design plant, and these were weighted by the inverses of $\underline{u}_{e_{max}}$ and $\dot{\underline{u}}_{e_{max}}$ to reflect actuation limits. The MATRIX$_X$® "EXEC" file for engine–airframe interface subcontroller design is listed in Appendix 8.2.2.2. The singular values of the engine–airframe subcontroller were plotted.

4.6.3 Engine–Airframe Subcontroller Reduction

The H_∞ engine–airframe subcontroller was 16th–order before order reduction (6 engine states, 4 engine actuator states, 3 sensitivity weighting states, and 3 complementary sensitivity weighting states). This subcontroller was reduced to 4th order by truncating the internally balanced realization.

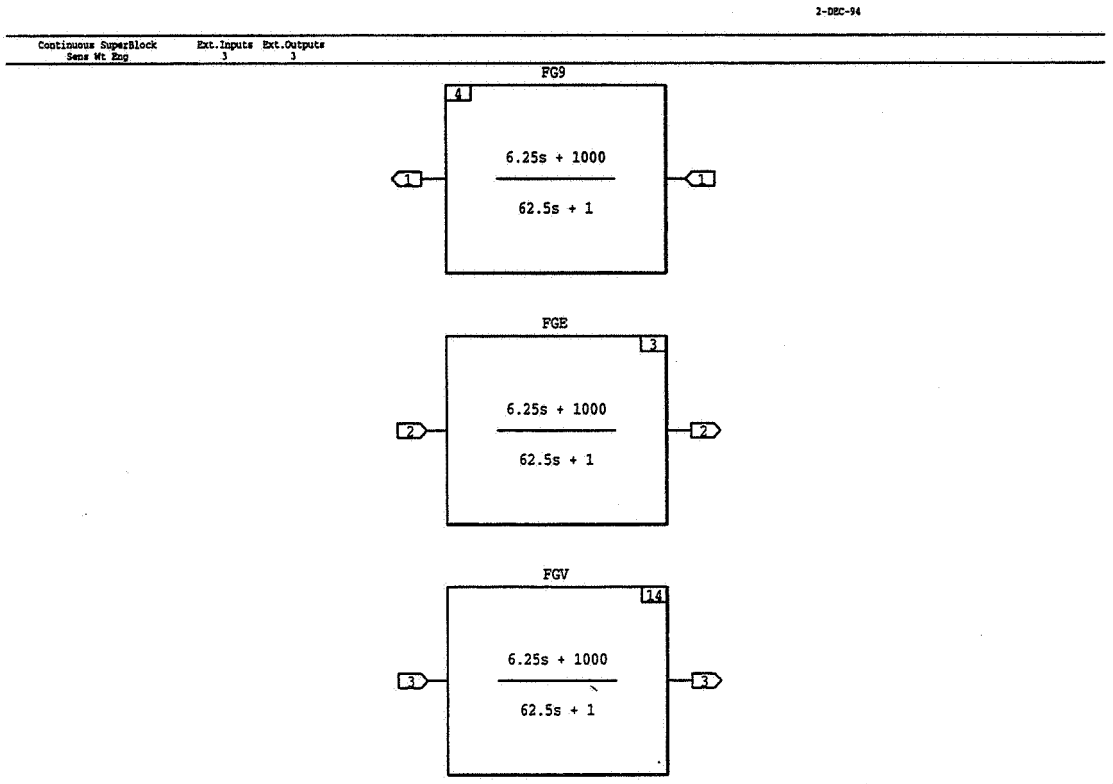


Figure 4.15 Engine Sensitivity Weighting Filters

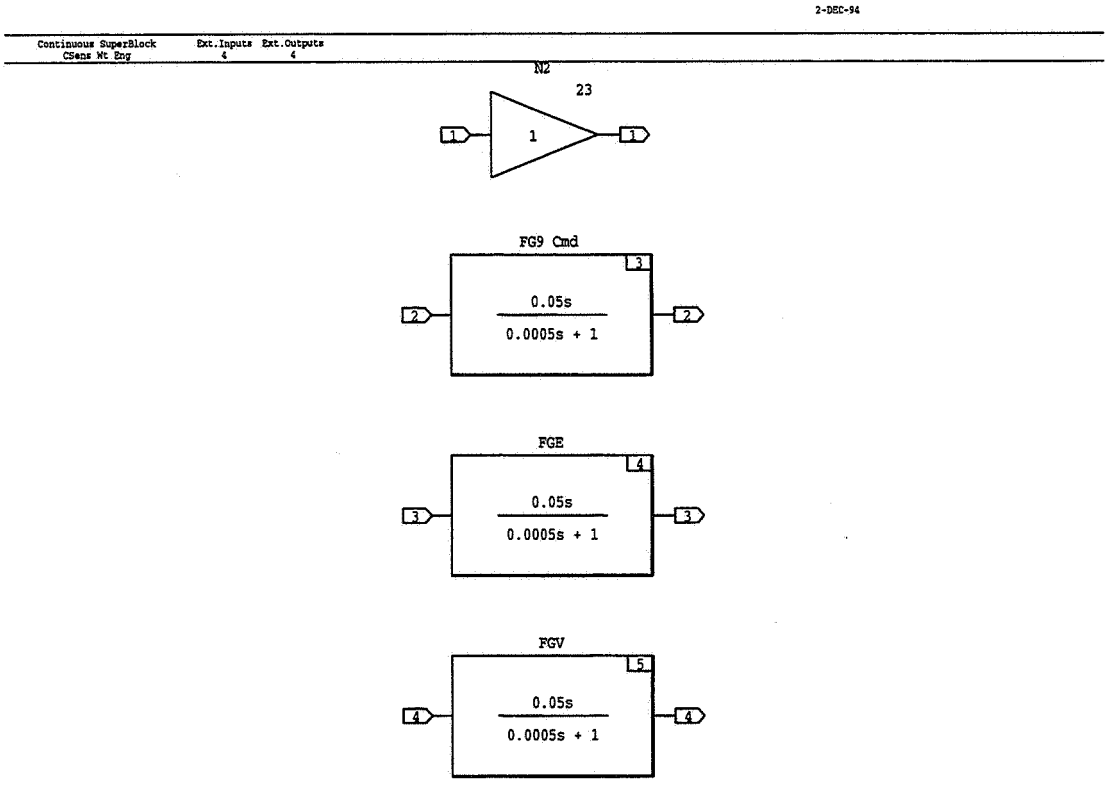


Figure 4.16 Engine Complementary Sensitivity Weighting Filters

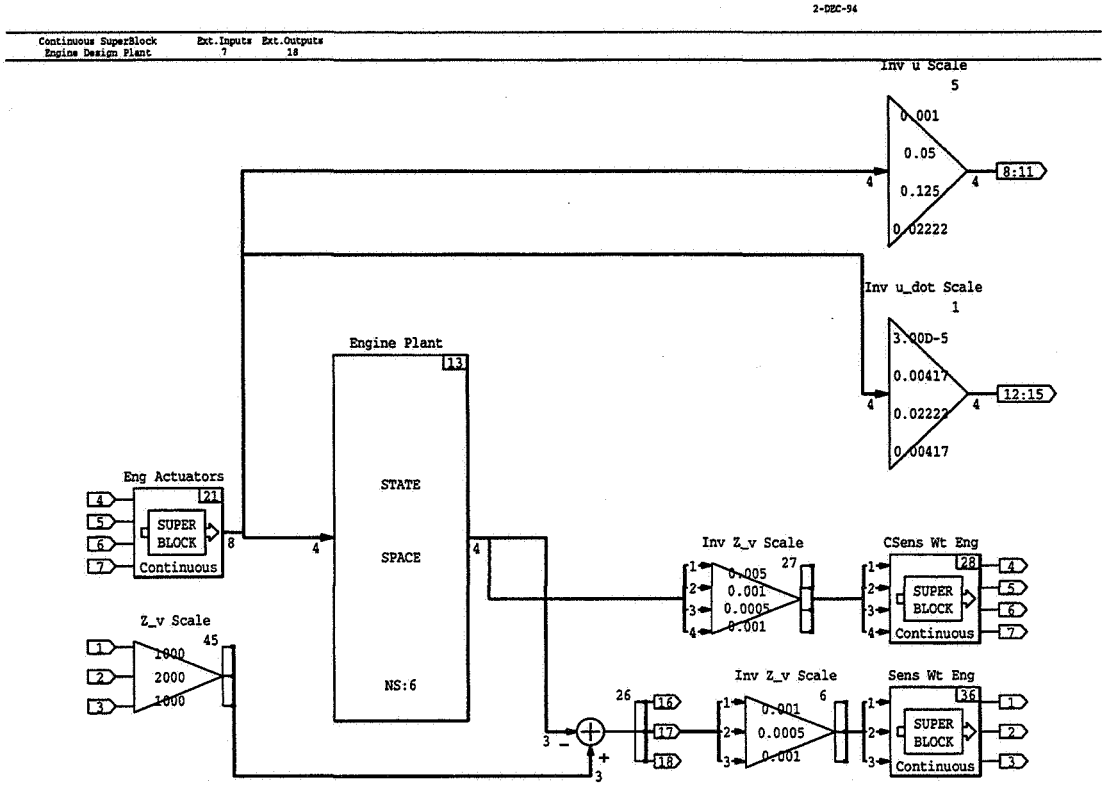


Figure 4.17 Engine Design Plant

The MATRIX$_X$® "EXEC" file for engine–airframe interface subcontroller order reduction is listed in Appendix 8.2.2.3. The incoming full–order engine–airframe subcontroller was scaled and its singular values computed and plotted. The subcontroller was balanced and truncated. The singular values of the reduced subcontroller were plotted, and the minimum and maximum singular values were compared with those of the full–order subcontroller. The reduced–order subcontroller was then rescaled. The singular values of the closed–loop engine–airframe system with the full–order and reduced subcontrollers were then plotted separately, and the minimum and maximum singular values were overplotted for comparison. Figure 4.18 shows the comparison of the minimum and maximum singular values for the original and reduced airframe subcontrollers. Figure 4.19 shows the minimum and maximum singular values of the closed–loop engine–airframe system with original and reduced engine–airframe subcontrollers.

Based on responses to step input commands in N2, FGV, FGE, and FG9, we elected not to reduce the engine–airframe subcontroller below 4th order, although NASA Lewis researchers elected to reduce to 3rd order.

4.6.4 Engine–Airframe Subcontroller Closed–Loop Analysis

The MATRIX$_X$® model for the engine–airframe closed–loop analysis is shown in Figure 4.20. Incoming $\underline{z}_{e_c}$ and $\underline{z}_{ea_c}$ values were commanded N2, FG9, FGE, and FGV values. The propulsion controller, $K^e(s)$, obtained by combining $K^e_e(s)$ and $K^e_{ea}(s)$, closed the engine loop.

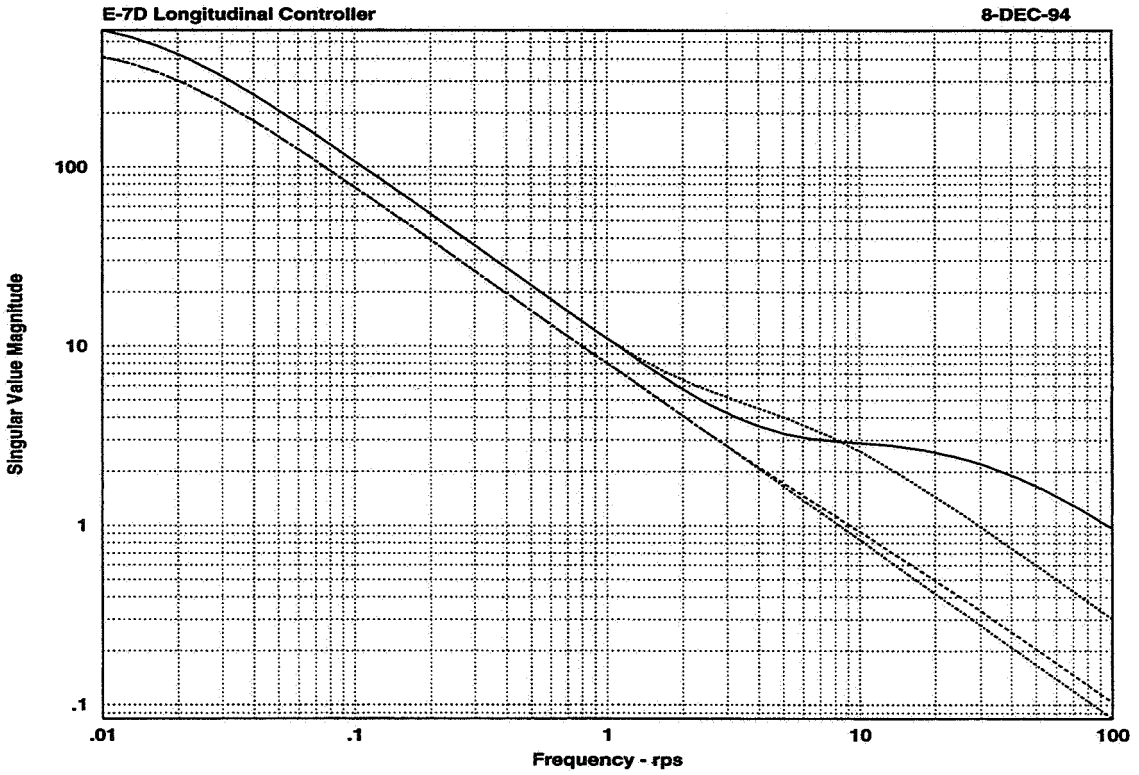


Figure 4.18 Minimum and Maximum Singular Values: Original and Reduced Engine–Airframe Subcontrollers

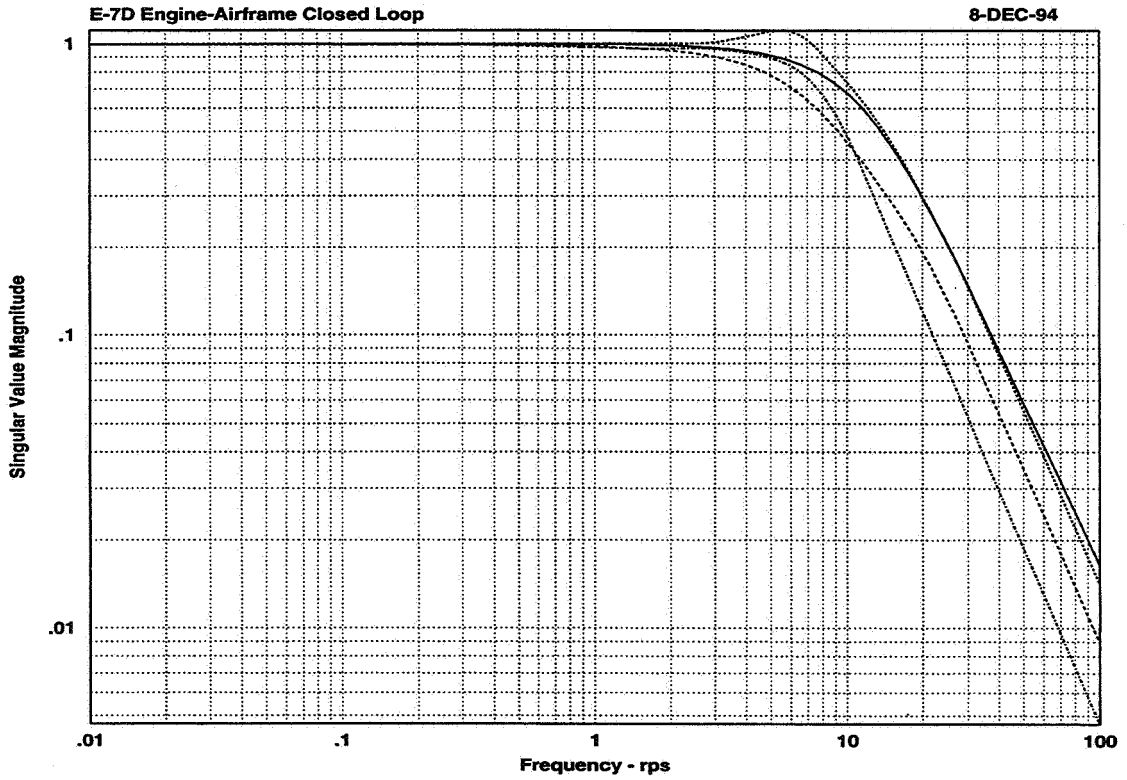


Figure 4.19 Minimum and Maximum Singular Values: Closed–Loop Engine–Airframe System with Original and Reduced Engine–Airframe Subcontrollers

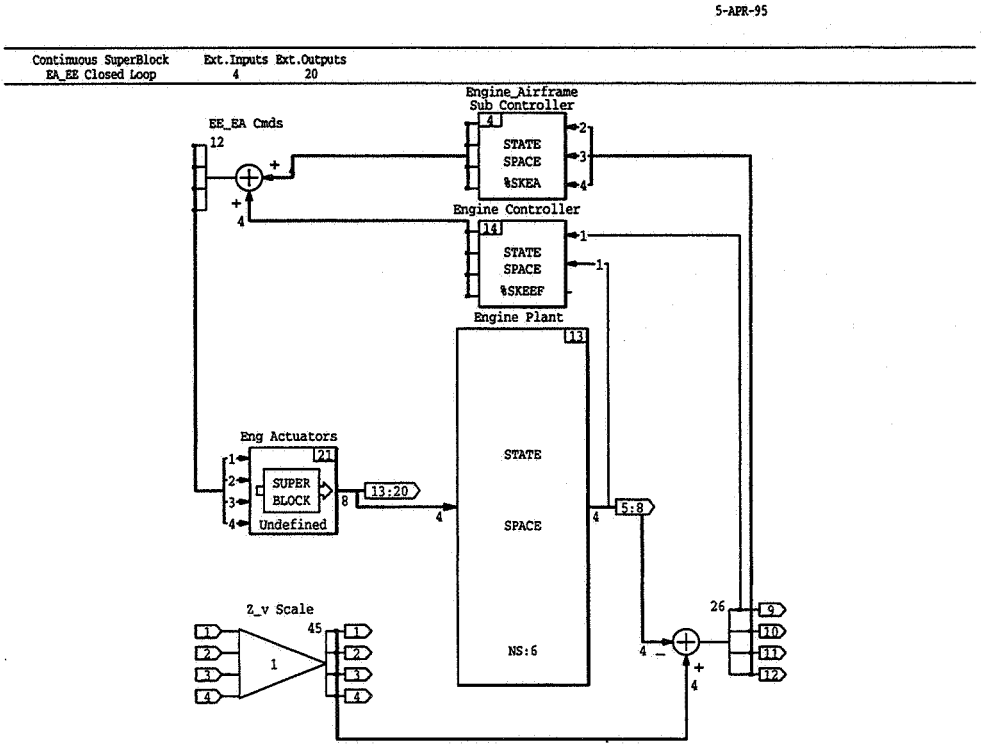


Figure 4.20 Engine–Airframe Closed–Loop Model

4.6.4.1 Engine–Airframe Subcontroller Closed–Loop Singular Value Analysis

The MATRIX$_X$® "EXEC" file for engine–airframe interface subcontroller closed–loop singular value analysis is listed in Appendix 8.2.2.4. This file provides for computing several sets of singular values: all inputs to all outputs, $\underline{z}_{e_c}$ and $\underline{z}_{ea_c}$ command inputs to $\underline{z}_{e_c}$–$\underline{z}_e$ and $\underline{z}_{ea_c}$–$\underline{z}_{ea}$ error outputs, $\underline{z}_{e_c}$ and $\underline{z}_{ea_c}$ inputs to $\underline{z}_e$ and $\underline{z}_{ea}$ outputs, and $\underline{z}_{e_c}$ and $\underline{z}_{ea_c}$ inputs to $\underline{u}_e$ and $\dot{\underline{u}}_e$ outputs. All maximum singular values were then plotted together for comparison.

4.6.4.2 Engine–Airframe Subcontroller Closed–Loop Simulation

The MATRIX$_X$® "EXEC" file for engine–airframe interface subcontroller closed–loop simulation is listed in Appendix 8.2.2.5. This file provides for plotting linear system response to unit step $\underline{z}_{e_c}$ and $\underline{z}_{ea_c}$ commands in N2, FG9, FGE, or FGV. The user is prompted for which of these four commands is to be simulated, and responses of $\underline{z}_{ea}$ and $\underline{u}_e$ were plotted. The MATRIX$_X$® "EXEC" file for closed–loop simulation of the propulsion controller with the reduced instead of full–order engine–airframe subcontroller is listed in Appendix 8.2.2.6.

4.6.5 Conclusions and Recommendations Regarding Engine–Airframe Subcontroller Design

Unlike the other subcontrollers, designing the engine–airframe subcontroller involved specifying and solving another H_∞ control design problem. We had some questions regarding how much the

order of the subcontroller should be reduced. Developing some guidelines for this could be extremely useful for others using IMPAC.

4.7 Airframe Subcontroller Design

4.7.1 Airframe Subcontroller Partitioning and Order Reduction

Figure 4.21 shows the MATRIX$_X$® model for airframe subcontroller extraction. (This figure

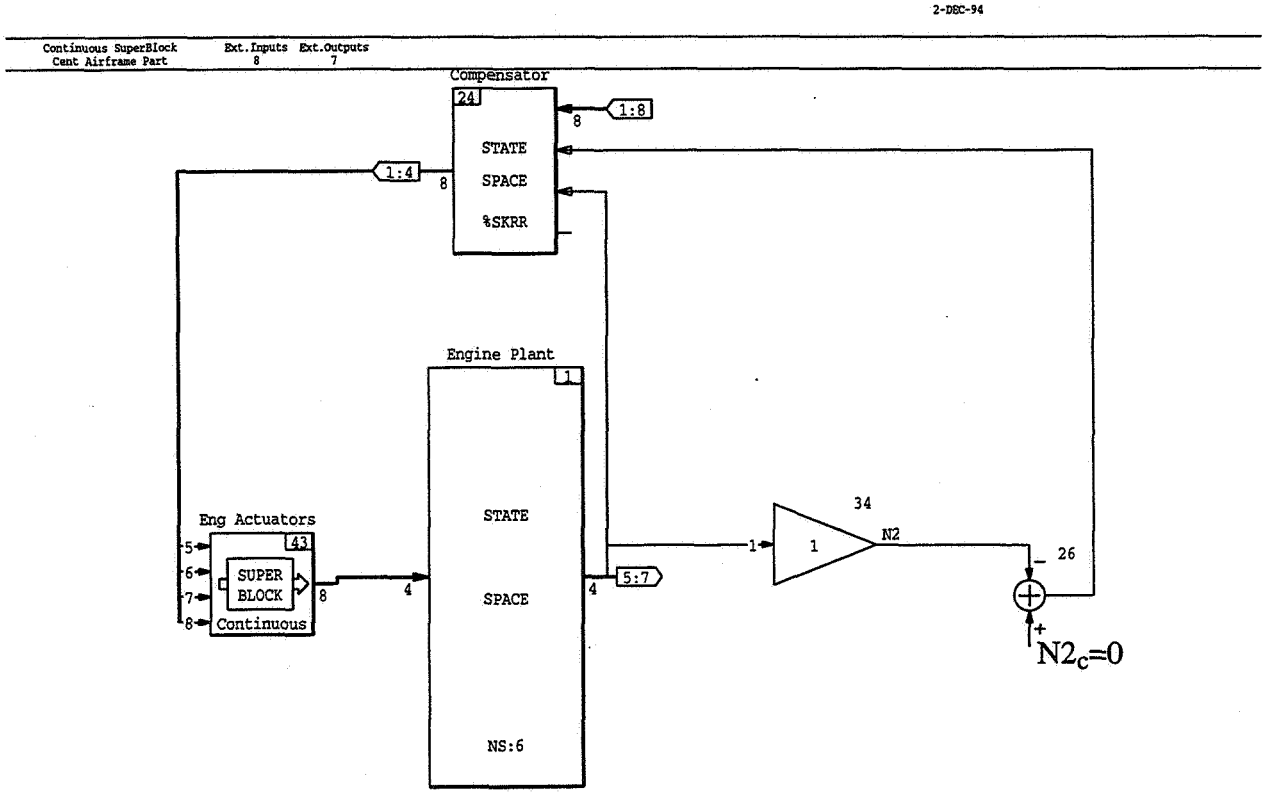


Figure 4.21 Centralized Controller Airframe Partition Model

corresponds to Figure 3.6 in the IMPAC overview discussion.) The eight external inputs to the compensator are $\underline{e}_a$ and $\underline{y}_a$, and the first four compensator outputs are $\underline{u}_a$. $\underline{z}_{ea}$ comprises three of the outputs from the engine plant, FG9, FGE, and FGV. N2 is the remaining output.

The MATRIX$_X$® "EXEC" file for airframe subcontroller partitioning is listed in Appendix 8.2.3. Using this file, the $[\underline{e}_a\ \underline{u}_a]^T \rightarrow [\underline{u}_a\ \underline{z}_{ea}]^T$ transfer function was extracted, scaled, and reduced by truncating the internally balanced realization. Each of the individual singular values of these original and reduced scaled airframe subcontrollers were plotted for comparison, as well as the maximum and minimum values. The reduced–order controller was then rescaled.

We chose not to reduce the airframe subcontroller below 12th order, although NASA Lewis researchers elected to reduce to 10th order. The 10th–order subcontroller gave notably better overall

response when the actuator models were taken out, but the 12th–order design gave notably better results when these models were included.

4.7.2 Lead Compensation

The lead compensation for each of the three gross thrust commands was chosen to be

$$K_i^{lead} = \frac{s + 4.5}{4.5} \frac{12}{s + 12}$$

resulting in an effective bandwidth of 12 rad/s for each of the $z_{ea_{des,i}} \rightarrow z_{ea_i}$ responses.

4.7.3 Conclusions and Recommendations Regarding Airframe Subcontroller Design

Obtaining the airframe subcontroller was straightforward, although our inclination was to have a slightly higher–order airframe subcontroller than in the NASA Lewis work. It was not difficult to incorporate leads as shown for the z_{ea} commands, but we wonder whether equivalent lead could not be obtained by adding appropriate requirements for the engine–airframe subcontroller design.

4.8 Partitioned Control Law Evaluation

4.8.1 Closed–Loop Analysis

The MATRIX$_X$® model for the closed–loop vehicle and partitioned subcontroller system is illustrated in Figure 4.22.

The MATRIX$_X$® "EXEC" file for closed–loop simulation of the vehicle with a partitioned controller is listed in Appendix 8.3. This file allows the user the option of simulating linear responses to unit step commanded Vv, Qv, flight path angle, or engine fan speed commands. The command, outputs, and error values were plotted, as were the airframe and engine control inputs, outputs, and interface variables. Figure 4.23–Figure 4.24 show responses in selected variables to a step flight path angle command with the centralized and partitioned controllers.

4.8.2 Conclusions and Recommendations Regarding Partitioned Control Law Evaluation

In our view, closed–loop step responses are more useful than singular value analyses in determining whether a controller design is truly suitable. Direct comparisons of the partitioned and original

Contimuous SuperBlock Ext.Inputs Ext.Outputs
Partitioned Closed Loop 4 37

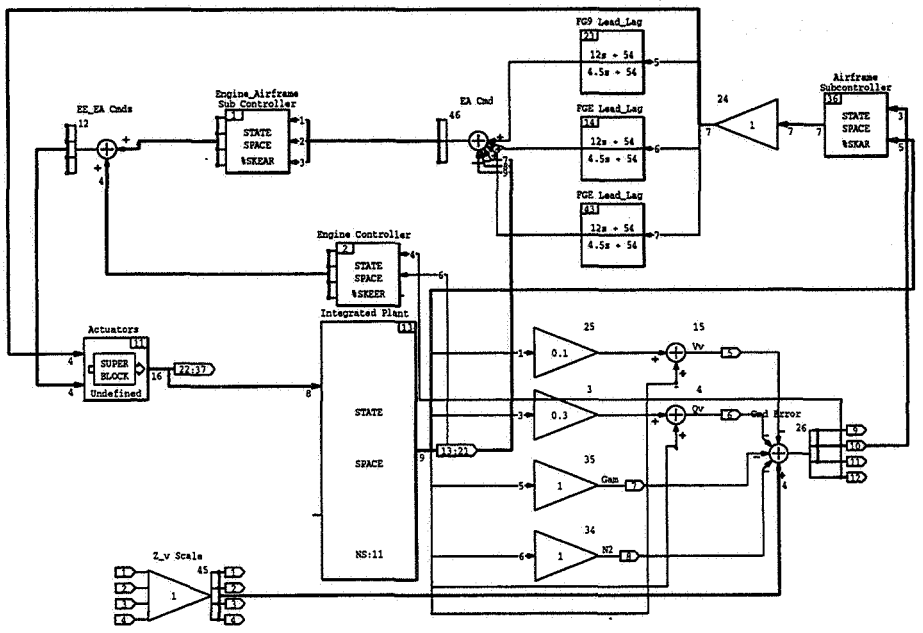


Figure 4.22 Partitioned Subcontroller Closed–Loop Model

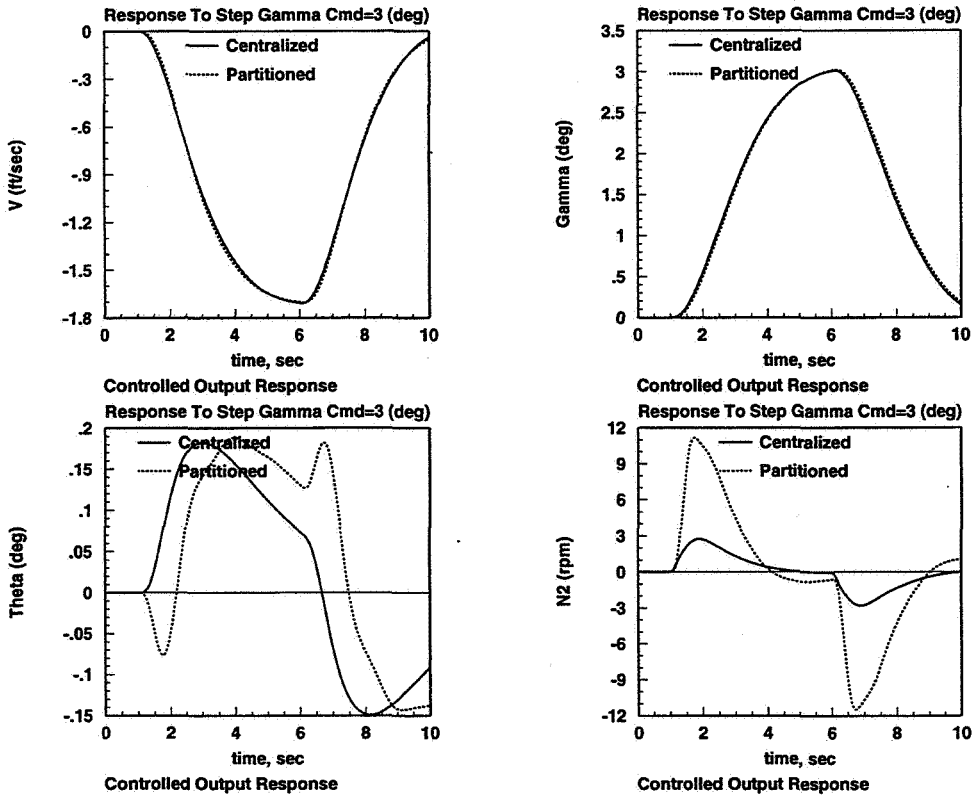


Figure 4.23 Closed–Loop System Response to Step Flight Path Angle Command

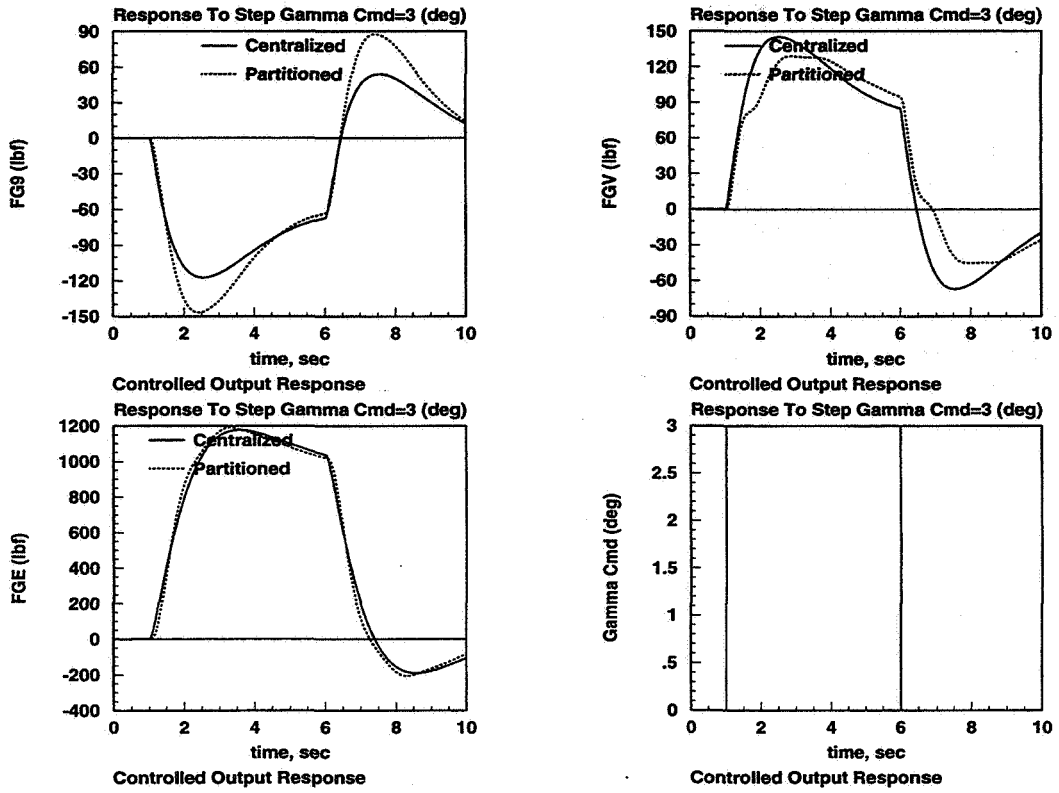


Figure 4.24 Closed–Loop System Response to Step Flight Path Angle Command (cont'd)

centralized controller were extremely useful and are recommended. After all the order reduction, it also seems advisable to monitor how the value of the H_∞ norm bound changes relative to the value of the original centralized controller.

4.9 IMPAC Methodology Conclusions

It was basically a straightforward process to replicate the NASA Lewis work applying the IMPAC design methodology to the E–7D STOVL aircraft, although significant time using and developing MATRIX$_X$® tools was required. This vehicle is a challenging application. Although we did not extensively evaluate the final partitioned engine, engine–airframe, and airframe linear point–condition controllers, they appear to function adequately in tracking step commands. In designing the engine–airframe and airframe subcontrollers, we might have preferred retaining higher–order controllers than the NASA Lewis researchers. However, our work met the objectives of allowing us to understand the IMPAC process and become more familiar with modern control design techniques.

We recognize that developing the command variables and measurement feedbacks and partitioning the commands and controls might have required a significant trial and error process. Developing command variables, tailored flight modes, and mode switch logic is a significant part of LMTAS control design. We recommend providing guidelines for this. We also recommend developing guidelines for choosing weighting matrices for the H_∞ design process and for controller order

reduction. We recommend considering the use of generalized controls and providing a separate control selector. Modifiying the IMPAC process to use generalized controls will be nontrivial.

We have looked briefly at the NASA Lewis subcontroller optimization work, which was initiated to improve the performance of the partitioned controllers. In this work, the global controller was partitioned by selecting desired subcontroller structures and performing a parameter optimization procedure. The subcontrollers were initialized employing the global controller results. The optimized performance index included controller transfer function and system tracking error terms. In light of the expected difficulty in choosing partitioning for the command and control variables, we wonder whether this optimization procedure could be expanded to cross the controller partitioning boundaries if necessary.

Numerous attempts have been made to implement multivariable control laws into truly nonlinear aircraft systems, for example, with the F–22 aircraft. These efforts are typically abandoned for more classical approaches (e.g., with F–22). Although multivariable control methods have received a tremendous amount of exposure in the literature, very little treatment has been given to practical design issues, such as operating on control power (actuator) limits, and then preventing integrator windup and prioritizing usage of the available control power under these conditions. These issues are critical for STOVL aircraft, since the aircraft is typically required to operate on rate and position actuator limits (usually, of the engine) during powered–lift flight.

While under contract to NASA Lewis Research Center, LMTAS developed a multivariable reconfiguration, antiwindup, and axis prioritization scheme that showed good promise in addressing these issues and could be used with any basic controllers. The scheme yielded acceptable control performance under control power–limited conditions during piloted motion–based simulation at NASA Ames. Although further development of this work is required, it should be of interest to many who are trying to apply multivariable methods.

We would like to see IMPAC applied to a further expanded integrated control problem. An airframe/ engine/ outer loop guidance problem, airframe/ propulsion/ thermal control problem, or airframe/ propulsion/ structural dynamics control problem would all be further good tests of the IMPAC design process.

5. IMPAC DESIGN TOOL

5.1 Current IMPAC Design Process Implementation

As described in this document, we implemented the IMPAC design process using different capabilities of the MATRIX$_X$® integrated control design tool; NASA Lewis has also used MATRIX$_X$® for its own IMPAC work. Functions in the basic and robust control modules and the System Build graphical system modeling tool were used. 19 different user–defined function macros were used; 14 were written at LMTAS and 5 were contributed directly by NASA Lewis. These macros were (to some degree) tailored specifically for the E–7D application. Most of these macros were sizeable and would require some time to recreate by a new user.

5.2 IMPAC Design Process Flow

In order to better isolate how an IMPAC design tool could work, Figure 5.1–Figure 5.5 show a detailed notional view of the IMPAC design process from the standpoint of user input and output and predefined functions to implement each distinct step of the design calculations. This diagram also shows where optional analysis calculations and offline design guidance documentation could be made available to the tool user. This diagram represents only limited experience with IMPAC and deserves further refinement, but it may serve as a good starting place for developing a software tool.

IMPAC involves numerous distinct groups of calculations. Compared with the current approach of using several detached macros or low–level calculations, tying all these together in a tool could result in considerable time savings to a potential user and less possibility for error.

5.3 General Desired Features of IMPAC Design Tool

We recommend that the IMPAC tool be menu–driven as much as possible and the user prompted for specific type–in information only when needed. Ideally, the tool would provide bookkeeping for what has been input and generate appropriate warnings when specific needed information is not present. Context–sensitive "intelligent" help should preferably always be available. In addition, design notes would be available in separate scrollable windows for certain design steps. The user would be notified that these are available and be able to develop and add additional design guidance documentation. The tool should have enough smarts to display plots, for example, in the most helpful manner—superimposed if possible, above/ below otherwise.

5.4 Re Graphical User Interface for IMPAC Design Tool

The Flight Control Design and Analysis group at LMTAS has implemented a graphical user interface (GUI) for our trim/ linearization/ simulation tool, ATLAS. This interface has been

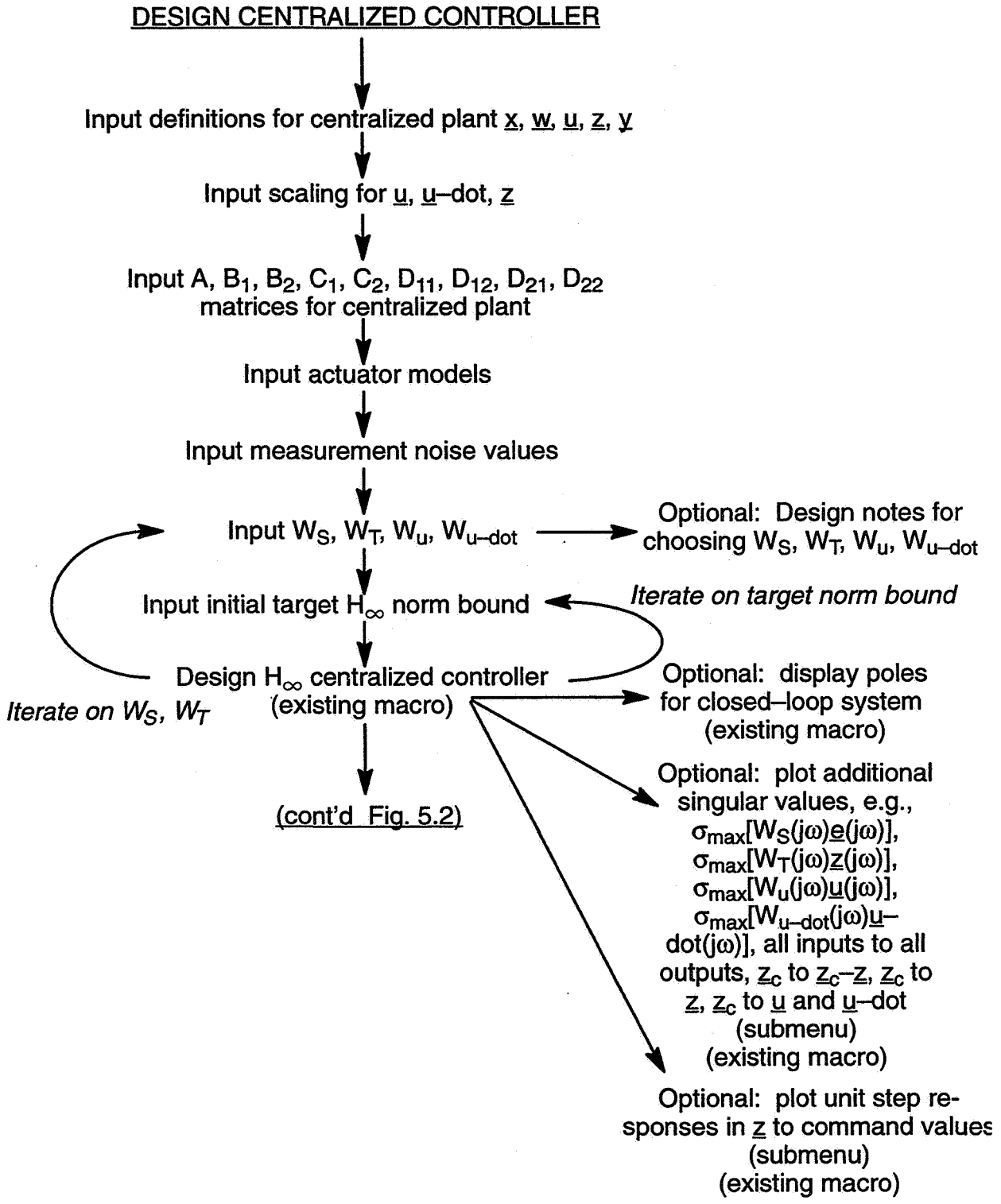


Figure 5.1 Suggested IMPAC Tool Input/ Output/ Calculation Flow (Sheet 1)

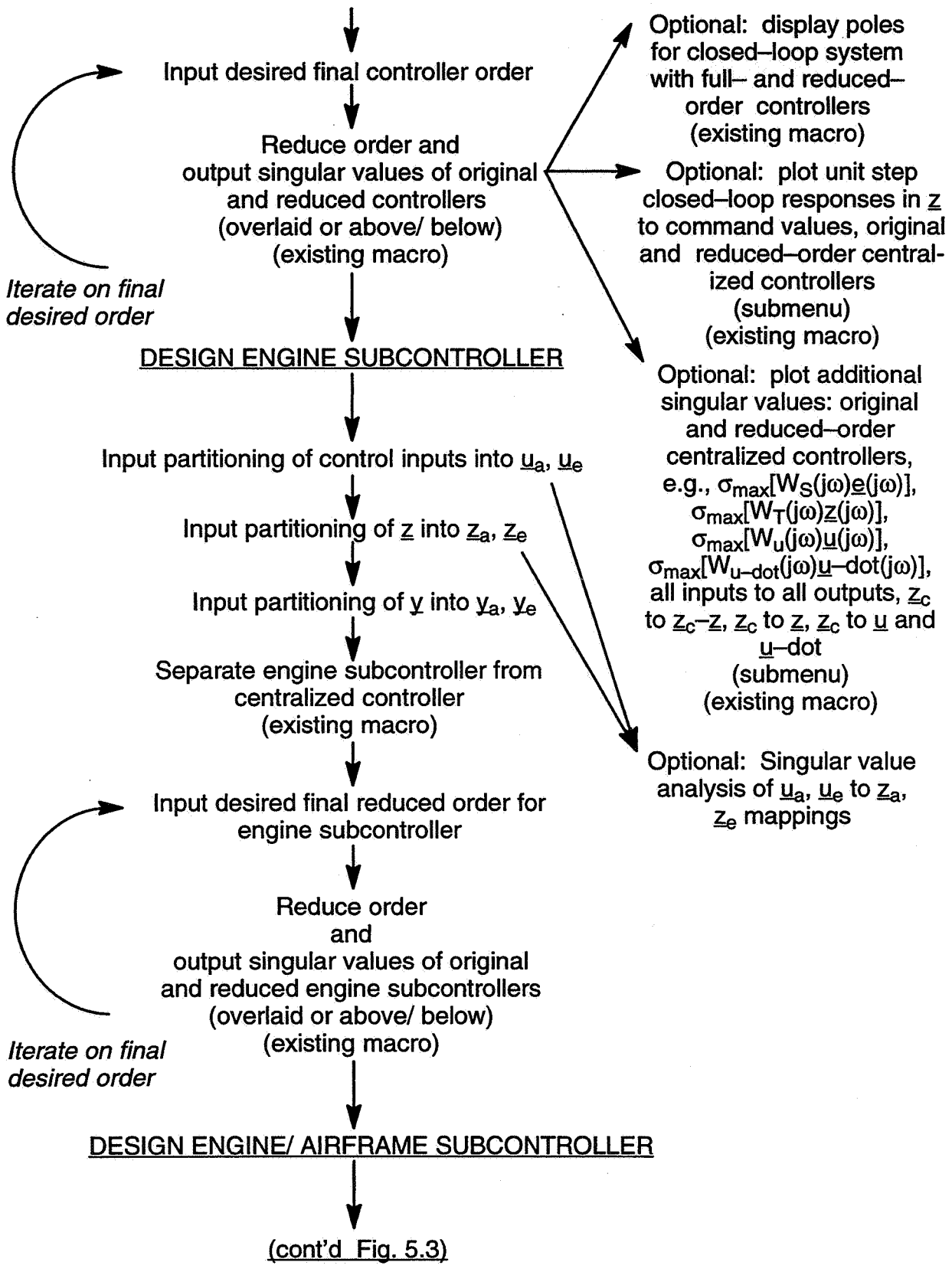


Figure 5.2 Suggested IMPAC Tool Input/ Output/ Calculation Flow (Sheet 2)

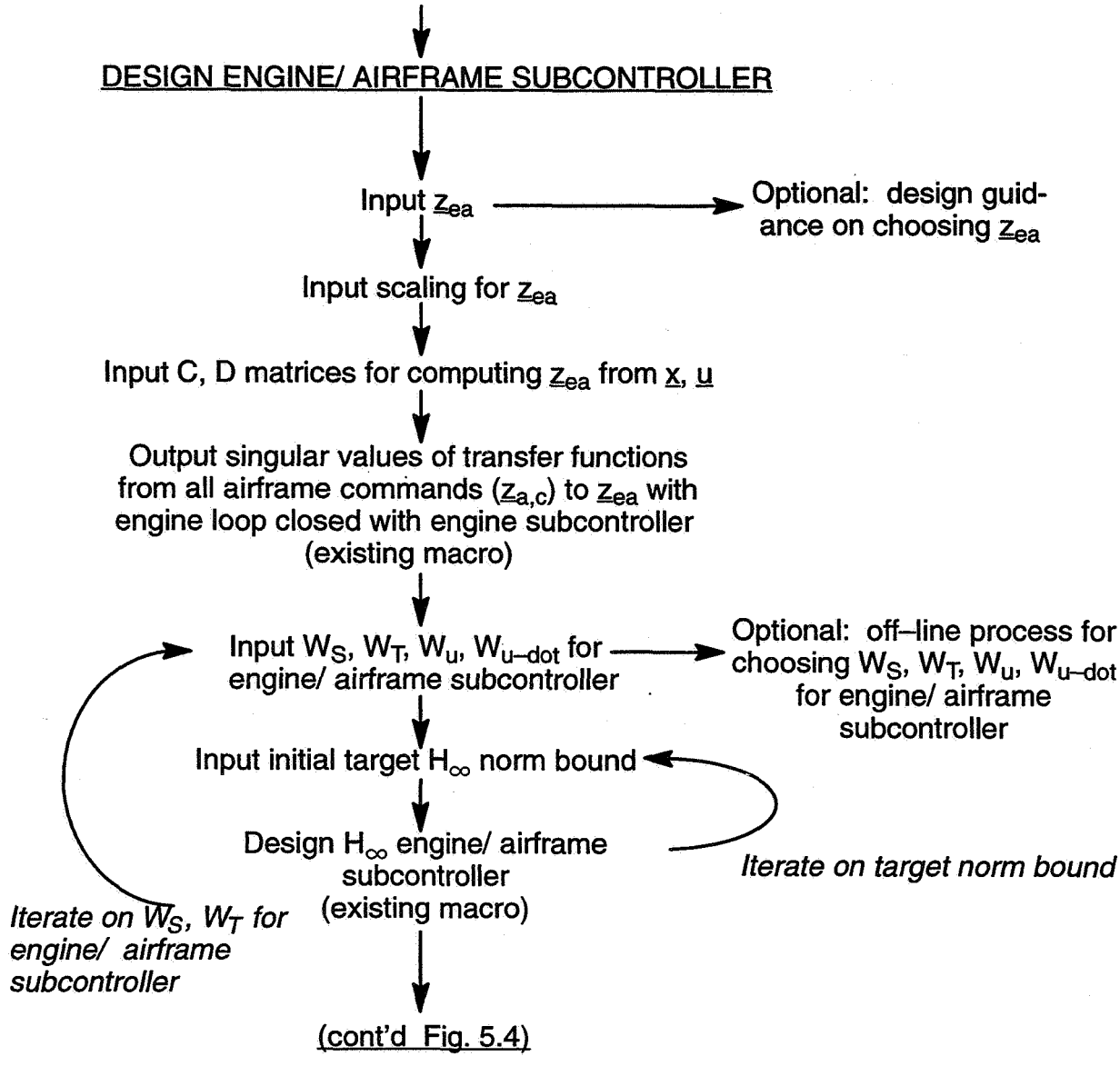


Figure 5.3 Suggested IMPAC Tool Input/ Output/ Calculation Flow (Sheet 3)

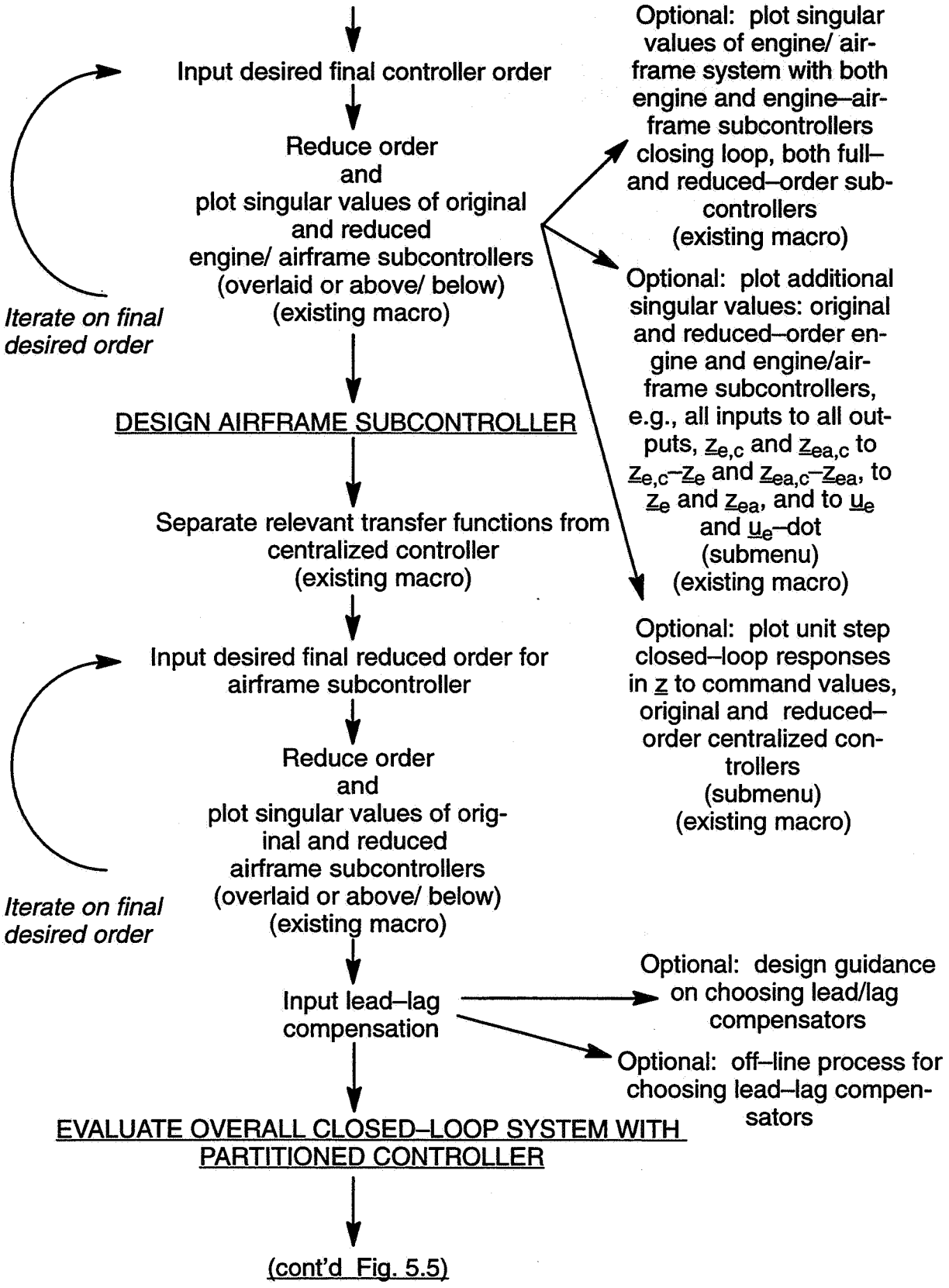


Figure 5.4 Suggested IMPAC Tool Input/ Output/ Calculation Flow (Sheet 4)

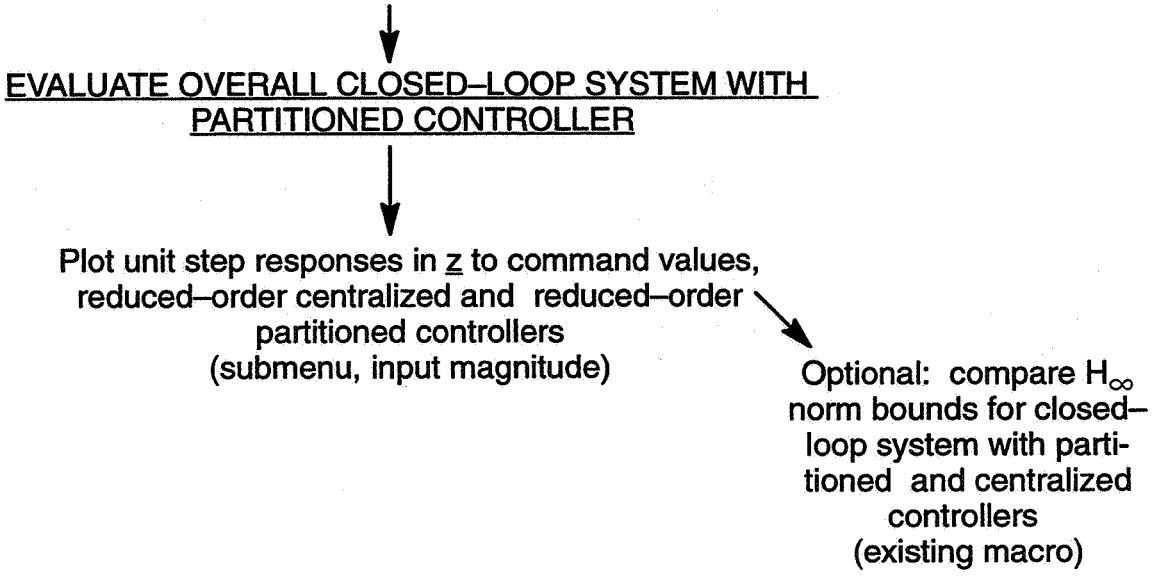


Figure 5.5 Suggested IMPAC Tool Input/ Output/ Calculation Flow (Sheet 5)

implemented on Pentium® personal computers, VAXstation™ and other Digital Equipment Corp. platforms, and Silicon Graphics platforms. Motif™ was used with X Window System™–type interface (or eXceed/NT™ for X Window System™ emulation on personal computer) to create this specific user interface.

The ATLAS graphical user interface has pull–down menus for basic commands as well as command–line prompts and user input echoing (Figure 5.6). Pop–up windows are used for file name input. Separate output windows are created when results of calculations are to be displayed.

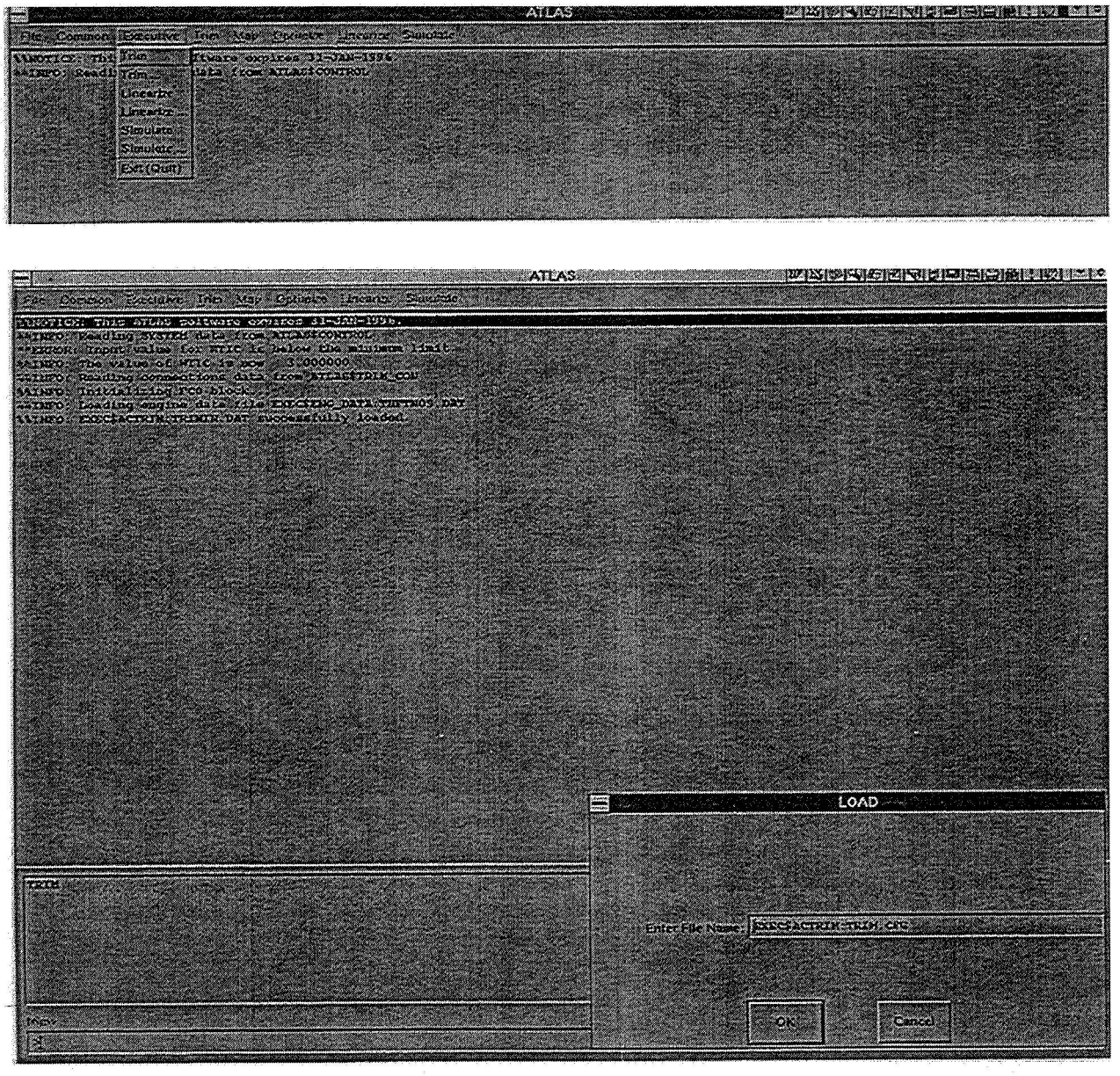


Figure 5.6 ATLAS User Interface

As with familiar Windows–type applications, certain menu items can be selected at certain points and not at others.

A similar menu–driven graphical user interface for IMPAC could be implemented with any of several available tools—Motif™, Xmath™ (with its new GUI interface builder), or MATLAB®. The main headings in the diagram of Figure 5.1–Figure 5.5 could be main menu headings, with the lower–level items being items on submenus or sub–submenus, as appropriate. We should conceptually be able to display calculation output and design guidance documentation using additional windows.

Our general approach in ATLAS is to have user input files with default filenames for the typical input information required. However, after these files have been loaded, the user also has the ability to deposit new information for any variable using the graphical user interface. Typically, the total input information is logically subdivided into several input files, and information is loaded only when needed. Some of these files are loaded by default; for others, the user is prompted with the default name and can input an alternate file name if needed.

Figure 5.7–Figure 5.11 show some notional menus for IMPAC. In principle, the IMPAC user should be able to input model vector and matrix information through a file or through a SystemBuild™ or SIMULINK™–type model, with model information input in state–space or transfer function form.

We should be able to program the graphical user interface available with Xmath™ and MATLAB® to be such that activating any pull–down menu item means that any set of the tool's primitive computations can be activated. Moreover, the script associated with the menu item should include the capability to activate any of the subtools of MATRIX$_X$®/ Xmath™ or MATLAB®. e.g., SystemBuild™ or SIMULINK™. In this way, the user could potentially input model information via these tools, if desired.

5.5 More on "Intelligent" Context–Sensitive Help

Having good online help and design guidance information will greatly impact the value of the IMPAC software tool. The online information should be sufficiently detailed for the most novice integrated propulsion/ airframe control designers. Help text should be available for every menu item available in the IMPAC tool.

Context–sensitive help information can be implemented in several ways. In existing Windows™ applications, clicking on any icon or any menu item brings up a new window titled with the icon or menu item name. In each such new window (as well as all top–level windows), "Help" is always listed as the far right–hand item on the top–level menu bar. Invoking this "Help" menu item brings up a new window with top–level help text associated with the last command, making it context–sensitive. The entire help information is also available at the same time by invoking menu items on the new help window menu bar.

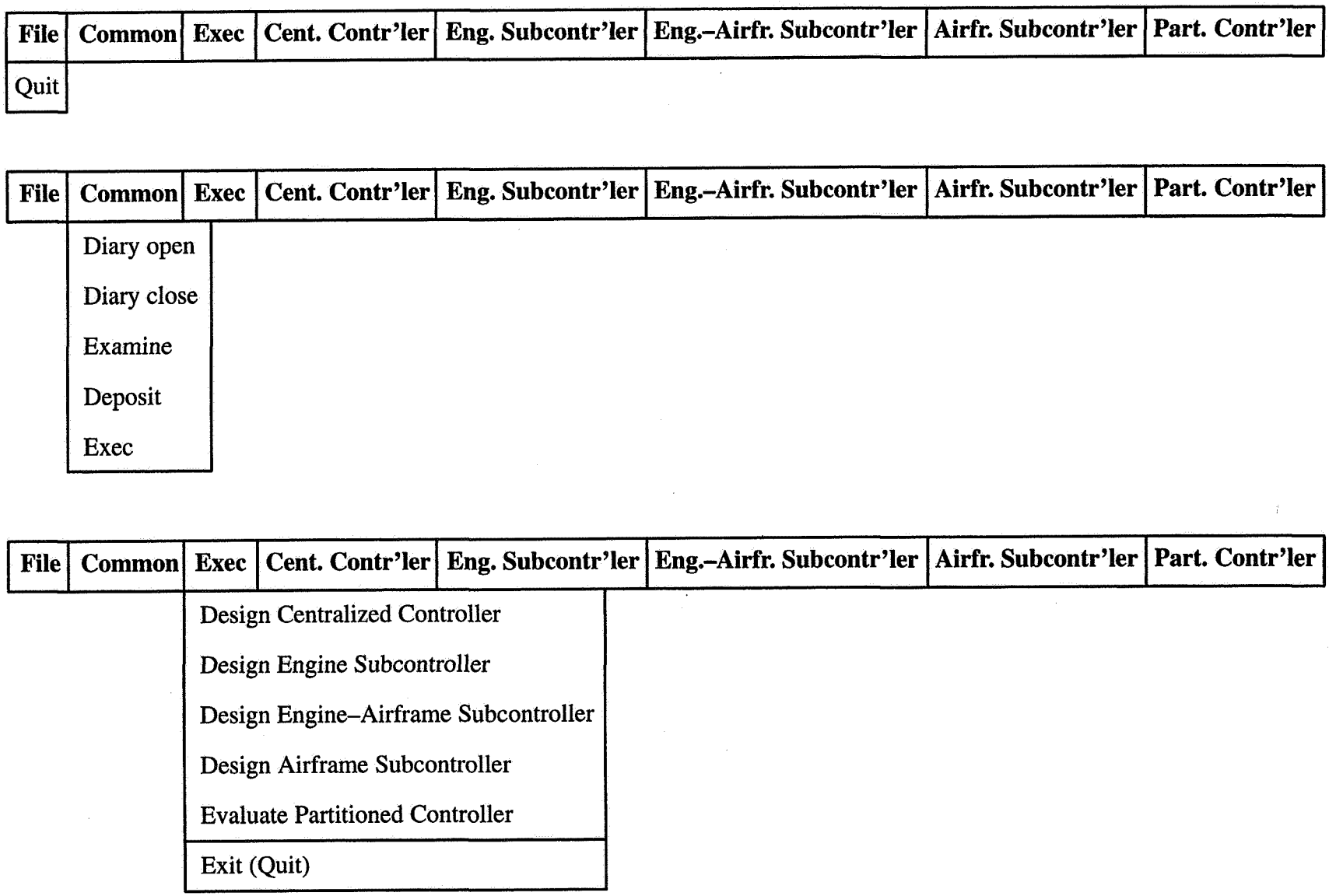


Figure 5.7 Notional IMPAC Menus (Sheet 1)

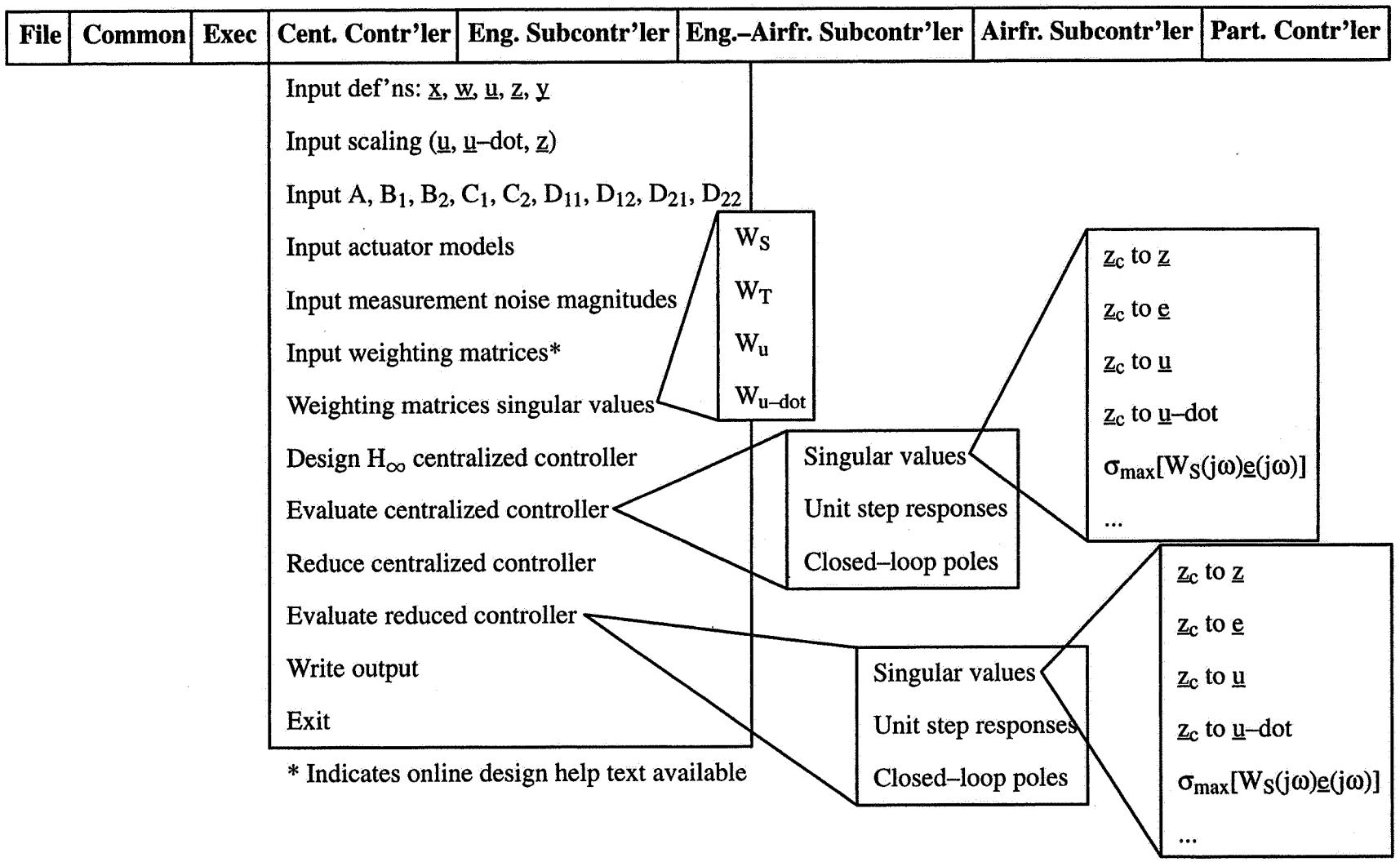


Figure 5.8 Notional IMPAC Menus (Sheet 2)

File	Common	Exec	Cent. Contr'ler	Eng. Subcontr'ler	Eng.–Airfr. Subcontr'ler	Airfr. Subcontr'ler	Part. Contr'ler

Input def'ns: $\underline{u}_a$, $\underline{u}_e$, $\underline{z}_a$, $\underline{z}_e$, $\underline{y}_a$, $\underline{y}_e$

Controllability of partitioning

Separate engine subc. from cent. controller

Reduce engine subcontroller

Write output

Exit

Figure 5.9 Notional IMPAC Menus (Sheet 3)

File	Common	Exec	Cent. Contr'ler	Eng. Subcontr'ler	Eng.–Airfr. Subcontr'ler	Airfr. Subcontr'ler	Part. Contr'ler

- Input def'n: $\underline{z}_{ea}$*
- Input scaling for $\underline{z}_{ea}$
- Input C, D matrices for computing $\underline{z}_{ea}$
- Plot $\underline{z}_{a,c}$ to $\underline{z}_{ea}$ transfer functions
- Input weighting matrices*
- Weighting matrices singular values
- Design H_∞ engine/ airframe subcontroller
- Reduce engine/ airframe subcontroller
- Evaluate reduced subcontroller
- Write output
- Exit

* Indicates online design help text available

Figure 5.10 Notional IMPAC Menus (Sheet 4)

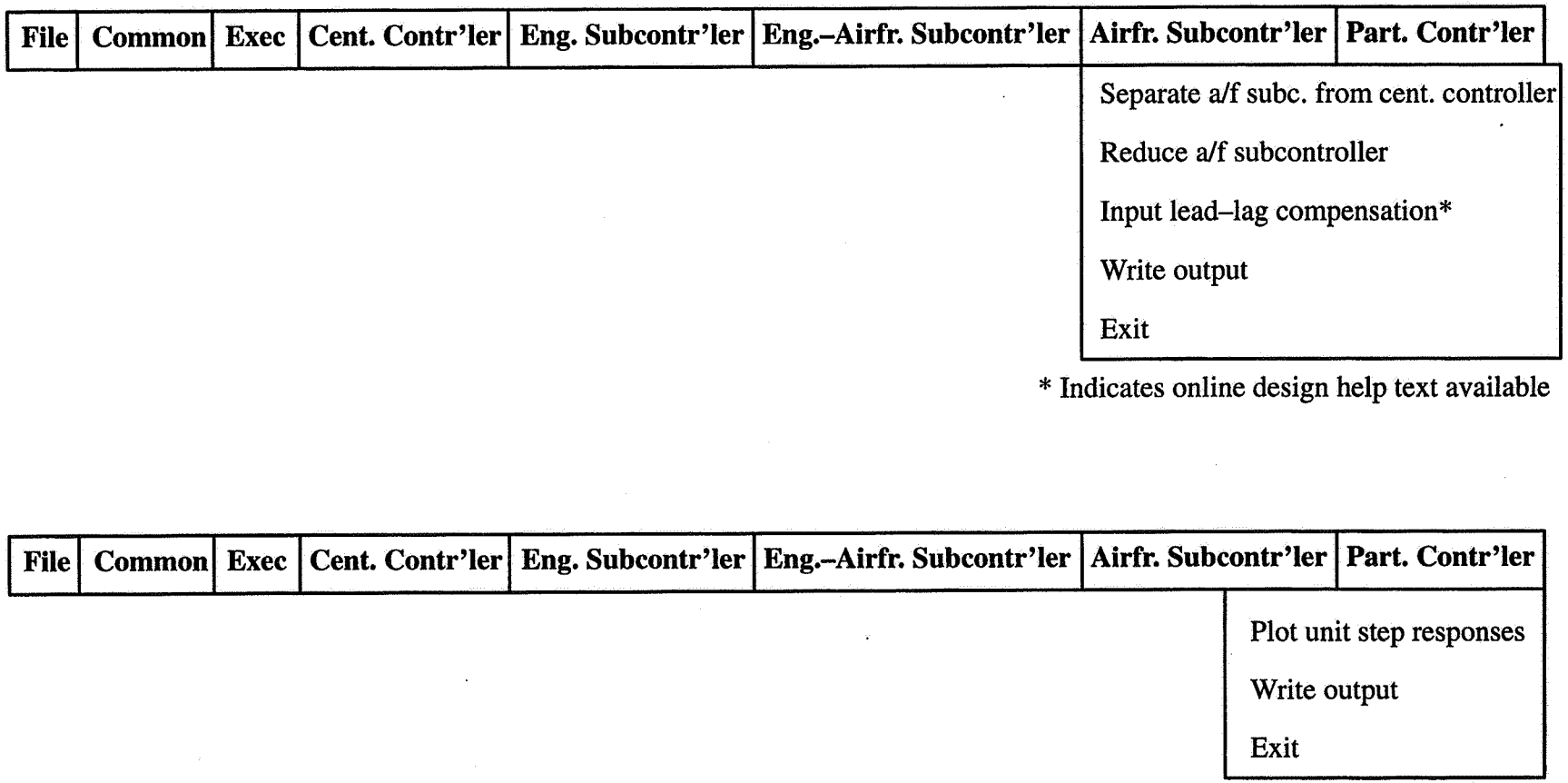


Figure 5.11 Notional IMPAC Menus (Sheet 5)

In our ATLAS tool, help information is accessible at any time by typing "help" at the command line prompt that is always available for use along with the menus. The help information listed is keyed to the last regular command, and the user enters a temporary help mode. All help information is also available, and any portion can be printed by typing the name of a command or item of interest.

Figure 5.12 shows a sample help text for the "Design H_∞ centralized controller" item on the "Centralized Controller" top–level menu.

Design H_∞ centralized controller

After input of the completely specified H_1 design problem, including definitions of states, noises and disturbances, control inputs, controlled variables, and measurement feedbacks, along with scaling, A, B_1, B_2, C_1, C_2, D_{11}, D_{12}, D_{21}, D_{22} matrices, actuator models, measurement noises, and weighting matrices, the iterative process of solving for the H_∞ centralized controller can begin. The user must specify an initial high value for the H_∞ "γ" value, the bound for the infinity norm. A macro will be used to invoke the series of steps involved in solving for the centralized controller such that H_∞ meets this value as an upper bound. If the γ value is achieved, a centralized controller has been solved for; if not, an error message will be returned. The γ value is then further decreased iteratively by the user until a centralized controller can no longer be solved for, and should then be increased again until a solution is obtained. A good indication that the original design objectives have been met is that the final H_∞ value is less than 1.

Figure 5.12 Example IMPAC Help Text

5.6 Tool Development and Usage Environment Trade–offs

Three separate environments for developing and using an IMPAC design tool will be compared:

1. "Custom" X Window System™/ C–Based (Figure 5.13)
2. MATRIX$_X$®/ Xmath™–Based (Figure 5.14)
3. MATLAB®–Based (Figure 5.15)

The IMPAC design process involves numerous matrix–oriented and primitive control design–oriented operations. Building a "custom" package would involve programming these operations. Motif™ is a commonly used graphical user interface builder, and the cost is very nominal ($100 for personal computer for eXceed/NT™ and Motif™). There is a significant initial learning curve for Motif™—it required six months for two people here to generate the first Motif™ interface

(for ATLAS), but it can be used to generate a very sophisticated interface. Motif™ needs to interface with C code, but can be implemented with software not written in C through a top–level C interface.

MATRIX$_X$® and MATLAB® are very comparable control design packages. Both have the requisite matrix–oriented and primitive control design and graphical output operations. Both have adequate tools to build graphical user interfaces, which appear to be roughly comparable in capability and maturity. Costs of the basic packages and the tool boxes that would be needed to perform the IMPAC calculations are very comparable and could be expected to remain so. MATLAB® is available for personal computer (including Apple), Sun, and Silicon Graphics platforms (VAX™ may no longer be supported)—and transferring MATLAB® code between platforms involves no changes to the code. MATRIX$_X$® is not available for Macintosh but is available for VAX™. MATLAB® has more following in the university setting, and MATRIX$_X$® more so in industry. Only Xmath™ supports autocoding in Ada, but both support autocoding in C and FORTRAN. Choosing between MATRIX$_X$® and MATLAB® for an IMPAC tool would require careful additional cost and capability analysis. Due to their similarity, developing versions of the tool for both (as with Xμ™ and μ–Tools™) would probably require considerably less than twice the time for either.

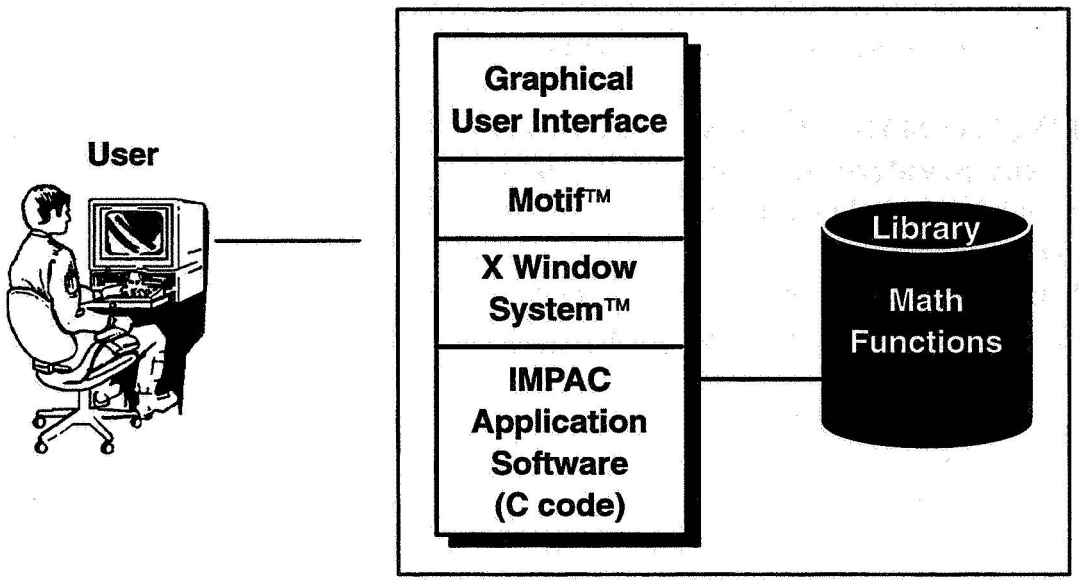


Figure 5.13 Custom IMPAC Software Tool

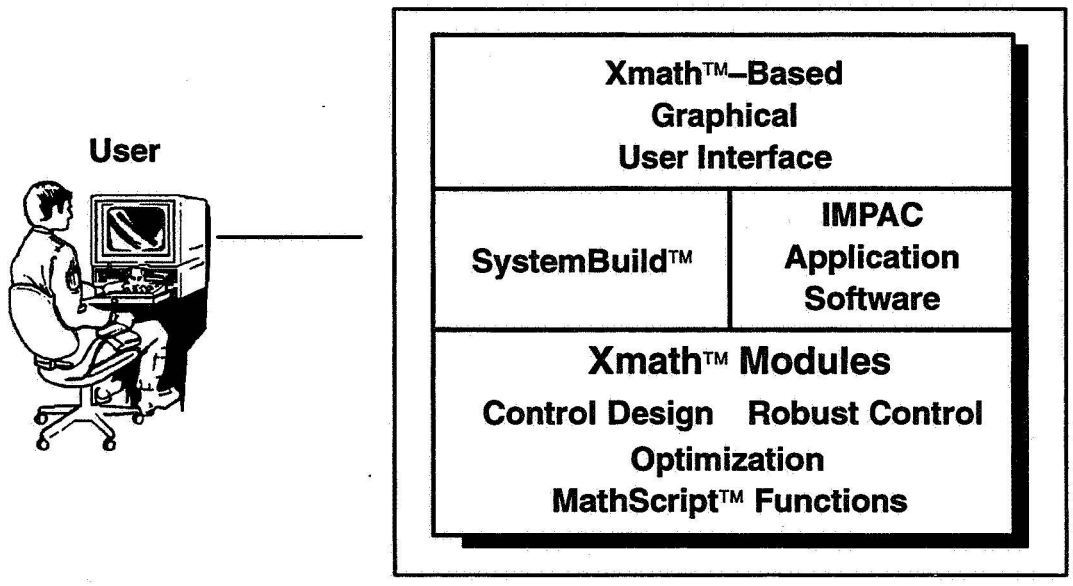


Figure 5.14 MATRIX$_X$® Xmath™–Based IMPAC Software Tool

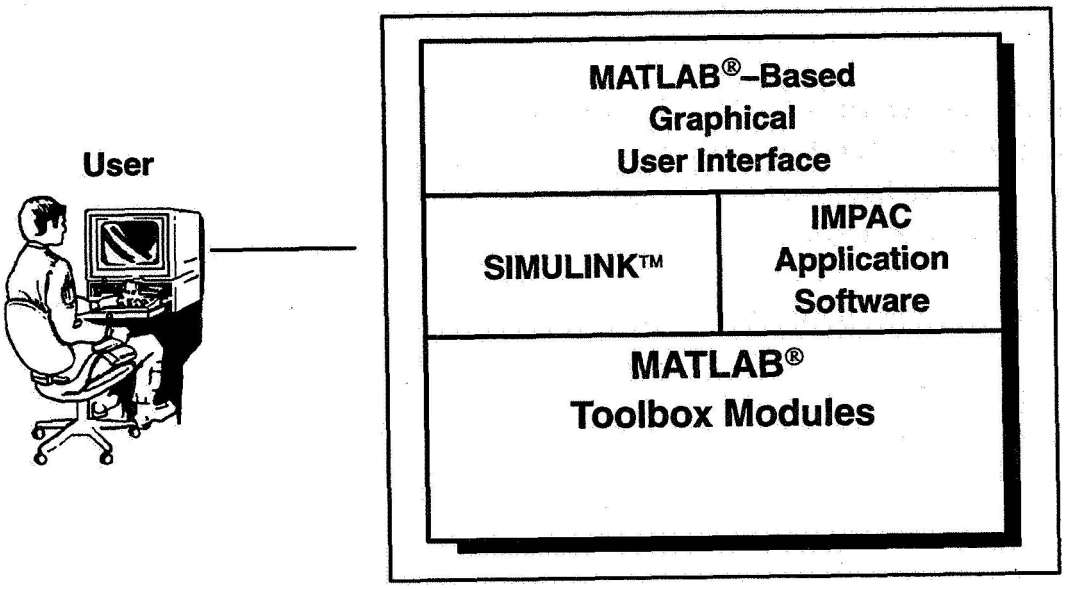


Figure 5.15 MATLAB®–Based IMPAC Tool

6. SUMMARY AND CONCLUSIONS

In this study, we exercised the IMPAC integrated airframe/ propulsion control design process using a model of the E–7D STOVL aircraft. Results obtained previously by NASA Lewis researchers were duplicated, and we have documented some comments and recommendations for each step of the design process based on our limited experience with it.

We can make several recommendations regarding the IMPAC design methodology:

—Develop guidelines for choosing command variables and control modes.
—Develop guidelines for system partitioning using, e.g., modal, Grammian, relative gain array, singular value, or optimization tools.
—Develop guidelines for choosing H_∞ weighting matrices.
—Develop guidelines for order reduction.
—Consider the use of generalized controls (using a control selector to map commands to physical controls).

The IMPAC design process is a multistep process, and a tool to automate it and provide design guidance could be extremely helpful. Developing such a tool would be straightforward, particularly using one of the integrated control design packages currently available, e.g., MATLAB® or MATRIX$_X$®, and their graphical interface building capabilities. We have diagrammed such a tool from the standpoint of suggested user input and output and predefined functions to implement distinct steps of the design calculations and drawn some notional menus for an IMPAC tool. We have provided notional menus to implement the IMPAC design process. Help text should be available for every menu item available in the IMPAC tool, and we have summarized good ways of implementing help information. Useful optional calculations and opportunities for offline design guidance were also indicated.

We recommend that an IMPAC tool be developed for both MATLAB® and MATRIX$_X$®. Both are very widely used in the control design community. Building a custom tool has no significant advantages in cost or usefulness.

7. REFERENCES

[1] Garg, Sanjay, Ouzts, Peter, Lorenzo, Carl, and Mattern, Duane, "IMPAC—An Integrated Methodology for Propulsion and Airframe Control," American Control Conference, Boston, MA, June 1991.

[2] Garg, Sanjay, "Robust Integrated Flight/ Propulsion Control Design for a STOVL Aircraft Using H–infinity Control Design Techniques," *Automatica*, Vol. 29, No. 1., pp. 129–145, 1993.

[3] Garg, Sanjay, "Partitioning of Centralized Integrated Flight/ Propulsion Control Design for Decentralized Implementation," *IEEE Transactions on Control Systems Technology*, Vol. 1, No. 2, June 1993, pp. 93–100.

[4] Shaw, P.D., Rock, S.M., and Fisk, W.S., "Design Methods for Integrated Control Systems," Report AFWAL–TR–88–2061, Wright Patterson AFB, OH, June 1988.

[5] Mattern, D.L., Garg, S., and Bullard, R.E., "Integrated Flight/ Propulsion Control System Design Based on a Decentralized, Hierarchical Approach," Paper 89–3519, AIAA Guidance, Navigation, and Control Conference, Boston, MA, Aug. 1989.

[6] Smith, K.L., "Design Methods for Integrated Control Systems, Report AFWAL–TR–86–2103, Wright Patterson AFB, OH, Dec. 1986.

[7] Garg, S., Mattern, D.L., and Bullard, R.E., "Integrated Flight/ Propulsion Control System Design Based on a Centralized Approach," *Journal of Guidance, Control, and Dynamics*, Jan.–Feb. 1991, pp. 107–116.

[8] Garg, Sanjay, "Controller Partitioning for Integrated Flight/ Propulsion Control Implementation," American Control Conference, Chicago, June 24–26, 1992 (also NASA TM–105804).

[9] Adibhatla, S., Cooker, P., Pajakowski, A., Romine, B., Virnig, J., and Bodden, D., "STOVL Controls Technology, Vol. I, Integrated Flight/Propulsion Control Design," NASA Contractor Report 195361, July 1994..

8. APPENDIX—MATRIX$_X$® Files Listing

8.1 MATRIX$_X$® Code – Centralized Controller Design

8.1.1 H∞ Controller Synthesis

MATRIX$_X$® "EXEC" files for centralized H_∞ controller solution:

```
//
//  Solve H-infinity Centralized Design
//
define 'sg_hinf.mtx'
resize('sstack',300000);
sim('analyze/Design Plant');
[sh,nsh] = lin(0);
nshe=[nsh 24 12];
inquire gam_in 'Enter initial guess of gamma: ';
//[sk,nsk,s_hew,ns_hew] = hinf_contr(sh,nshe,gam_in);
[sk,nsk,s_hew,ns_hew] = sg_hinf(sh,nshe,gam_in);
[omega,sv_hew] = svplot(s_hew,ns_hew,.01,100.);
msv_hew = 10**(max(sv_hew)/20),
display('*** Rerun if gamma is much different from initial guess ***');
clear msv_hew gam_in omega sv_hew;
```

8.1.2 Centralized Controller Reduction

MATRIX$_X$® "EXEC" file for centralized controller reduction (STOVL_CREDS.MTX):

```
//
//  Reduce Centralized Controller
//
// Inputs:
//           SK    - NSK x NSK Controller Matrix
//           NSK   - Order of Controller Matrix (SK)
//
// Outputs:
//           SKRR  - NSKRR x NSKRR Reduced Order Controller Matrix
//           NSKRR - Order of RO Controller Matrix (SKRR)
//
resize('sstack',300000);
//
//  Analyze Full Order Controller
//
Plot('hold name/E-7D Longitudinal Full Order Controller/ date');
[omega,sv4_kf] = svplot(sk,nsk,.01,100);
mxsv_kf = sv_kf(:,1);
mnsv_kf = sv_kf(:,8);
pause;
//
//  Reduce Controller
```

```
//
skm = modal(sk,nsk);
[skr,nskr] = mreduce(skm,nsk,[1:22]);
[skrb,sigrb,tb] = balance(skr,nskr);
[skrr,nskrr] = mreduce(skrb,nskr,[1:10]);
//
//  Analyze Reduced Order Controller
//
Plot('hold name/E-7D Longitudinal Red. Order Controller/ date');
[omega,sv_kr] = svplot(skrr,nskrr,.01,100);
mxsv_kr = sv_kr(:,1);
mnsv_kr = sv_kr(:,8);
pause;
//
//  Plot Comparison of Controller Singular Values
//
Plot('hold grid5 xlab/Frequency - rps/ color=1');
Plot('hold name/E-7D Longitudinal Controller/ date');
Plot('hold ylab/Singular Value Magnitude - dB/ logx');
Plot(omega,[mxsv_kf mxsv_kr mnsv_kf mnsv_kr]),
hard;
Pause;
//
//  Analyze Closed Loop with Full Order Controller
//
sim('analyze/Closed Loop');
[sclf,nsclf] = lin(0);
sclsv = [sclf(1:nsclf,1:nsclf+4);sclf(nsclf+5:nsclf+8,1:nsclf+4)];
Plot('hold name/E-7D Longitudinal Full Order Closed Loop/ date');
[omega,sv_clf] = svplot(sclsv,nsclf,.01,100);
pause;
mxsv_clf = sv_clf(:,1);
mnsv_clf = sv_clf(:,4);
//
//  Analyze Closed Loop with Reduced Order Controller
//
sim('analyze/Closed Loop Red');
[sclr,nsclr] = lin(0);
sclsv = [sclr(1:nsclr,1:nsclr+4);sclr(nsclr+5:nsclr+8,1:nsclr+4)];
Plot('hold name/E-7D Longitudinal Reduced Closed Loop/ date');
[omega,sv_clr] = svplot(sclsv,nsclr,.01,100);
mxsv_clr = sv_clr(:,1);
mnsv_clr = sv_clr(:,4);
pause;
//
//  Plot Comparison of Closed Loop Singular Values
//
Plot('hold grid5 xlab/Frequency - rps/ color=1');
Plot('hold name/E-7D Longitudinal Closed Loop/ date');
opts = 'ylab/Singular Value Magnitude - dB/ logx';
Plot(omega,[mxsv_clf mxsv_clr mnsv_clf mnsv_clr],opts),
hard;
pause;
clear mxsv_clf mxsv_clr mnsv_clf mnsv_clr sv_clf sv_clr sclsv;
clear mxsv_kf mxsv_kr mnsv_kf mnsv_kr sv_kf sv_kr;
```

```
clear scalein scaleout omega sk1 skrr1 skm skr nskr;
clear skrb sigrb tb;
```

8.1.3 Closed–Loop Analysis

8.1.3.1 Closed–Loop Linear Simulation of Centralized Controller Plus Plant

MATRIX$_X$® "EXEC" file for closed–loop linear simulation (STOVL_CLSIM.MTX):

```
//
//  Analyze and Linearize Closed Loop System
//
resize('sstack',300000);
resize('rstack',150000);
u_null = 0*[0:1:1000]';
for i=1:101,u_step(i)=0;end,
for i=102:601,u_step(i)=1;end,
for i=602:1001,u_step(i)=0;end,
//
sim('analyze/Closed Loop');
[scl,nscl] = lin(0);
//sim('analyze/Closed Loop Red');
//[sclr,nsclr] = lin(0);
//
//  Compute STEP Time Histories
//          12345678901234567890123456789
  smenu = ['      SIM MENU          ';...
           '   Velocity Cmd         ';...
           '     Pitch Cmd          ';...
           '    Flight Path Cmd     ';...
           ' Engine Fan Speed Cmd   ';...
           '       EXIT             '];
    sopt = MENU(smenu,1);...
    If  sopt = 1,...
      display('*** Running Linear Simulation ***'),...
      u_cmd = [u_step u_null u_null u_null];...
      [t,y_cmd] = lsim(scl,nscl,u_cmd,.01);...
    Elseif  sopt = 2,...
      display('*** Running Linear Simulation ***'),...
      u_cmd = [u_null u_step u_null u_null];...
      [t,y_cmd] = lsim(scl,nscl,u_cmd,.01);...
    Elseif  sopt = 3,...
      display('*** Running Linear Simulation ***'),...
      u_cmd = [u_null u_null 3*u_step u_null];...
      [t,y_cmd] = lsim(scl,nscl,u_cmd,.01);...
    Elseif  sopt = 4,...
      display('*** Running Linear Simulation ***'),...
      u_cmd = [u_null u_null u_null u_step];...
      [t,y_cmd] = lsim(scl,nscl,u_cmd,.01);...
    End,...
//
//  Plot Results
//
Plot('hold grid5 xlab/Time (Seconds)/ color=1');
Plot('hold name/E-7D Longitudinal Simulation/ date');
```

```
//
opts = 'ylab/Vv Cmd|Qv Cmd|Gamma Cmd|N2 Cmd/ strip';
Plot(t,[y_cmd(:,1) y_cmd(:,2) y_cmd(:,3) y_cmd(:,4)],opts),
pause;
hard;
//
opts = 'ylab/Vv Out|Qv Out|Gamma Out|N2 Out/ strip';
Plot(t,[y_cmd(:,5) y_cmd(:,6) y_cmd(:,7) y_cmd(:,8)],opts),
pause;
hard;
//
opts = 'ylab/Vv Error|Qv Error|Gamma Error|N2 Error/ strip';
Plot(t,[y_cmd(:,9) y_cmd(:,10) y_cmd(:,11) y_cmd(:,12)],opts),
pause;
hard;
//
opts = 'ylab/Velocity|V dot|Theta|P Rate|Gamma/ strip';
Plot(t,[y_cmd(:,13) y_cmd(:,14) y_cmd(:,15) y_cmd(:,16)
y_cmd(:,17)],opts),
pause;
hard;
//
opts = 'ylab/Rotor Sp (N2)|Noz Thr (FG9)|Ej Thr (FGE)|Vent Thr (FGV)/
strip';
Plot(t,[y_cmd(:,18) y_cmd(:,19) y_cmd(:,20) y_cmd(:,21)],opts),
pause;
hard;
//
opts = 'ylab/dele|AQR|ANG79|ANG8/ strip left nodate';
Plot(t,[y_cmd(:,22) y_cmd(:,23) y_cmd(:,24) y_cmd(:,25)],opts),
//
opts = 'ylab/WF|A8|ETA|A78/ strip right noname';
Plot(t,[y_cmd(:,26) y_cmd(:,27) y_cmd(:,28) y_cmd(:,29)],opts),
pause;
hard;
//
opts = 'ylab/dele rate|AQR rate|ANG79 rate|ANG8 rate/ strip left nodate';
Plot(t,[y_cmd(:,30) y_cmd(:,31) y_cmd(:,32) y_cmd(:,33)],opts),
//
opts = 'ylab/WF rate|A8 rate|ETA rate|A78 rate/ strip right noname';
Plot(t,[y_cmd(:,34) y_cmd(:,35) y_cmd(:,36) y_cmd(:,37)],opts),
pause;
hard;
//
clear u_cmd y_cmd u_null u_step opts sopt smenu;
Plot('reset');
//\\print/queue=ps_chaos/delete matplot.ps;*
display('*** All Plots Complete! ***')
```

8.1.3.2 Closed–Loop Linear Simulation of Reduced–Order Centralized Controller Plus Plant

MATRIX$_X$® "EXEC" file for closed–loop linear simulation with reduced–order centralized controller (STOVL_CLSIMR.MTX):

```
//
//  Analyze and Linearize Closed Loop System
```

```
//
resize('sstack',300000);
u_null = 0*[0:1:1000]';
for i=1:101,u_step(i)=0;end,
for i=102:601,u_step(i)=1;end,
for i=602:1001,u_step(i)=0;end,
//
//sim('analyze/Closed Loop');
//[scl,nscl] = lin(0);
sim('analyze/Closed Loop Red');
[sclr,nsclr] = lin(0);
//
//  Compute STEP Time Histories
//          12345678901234567890123456789
  smenu = ['      SIM MENU          ';...
           '   Velocity Cmd         ';...
           '     Pitch Cmd          ';...
           '    Flight Path Cmd     ';...
           ' Engine Fan Speed Cmd   ';...
           '       EXIT             '];
    sopt = MENU(smenu,1);...
    If  sopt = 1,...
      display('*** Running Linear Simulation ***'),...
      u_cmd = [u_step u_null u_null u_null];...
      [t,y_cmd] = lsim(sclr,nsclr,u_cmd,.01);...
    Elseif  sopt = 2,...
      display('*** Running Linear Simulation ***'),...
      u_cmd = [u_null u_step u_null u_null];...
      [t,y_cmd] = lsim(sclr,nsclr,u_cmd,.01);...
    Elseif  sopt = 3,...
      display('*** Running Linear Simulation ***'),...
      u_cmd = [u_null u_null u_step u_null];...
      [t,y_cmd] = lsim(sclr,nsclr,u_cmd,.01);...
    Elseif  sopt = 4,...
      display('*** Running Linear Simulation ***'),...
      u_cmd = [u_null u_null u_null u_step];...
      [t,y_cmd] = lsim(sclr,nsclr,u_cmd,.01);...
    End,...
//
//  Plot Results
//
Plot('hold grid5 xlab/Time (Seconds)/ color=1');
Plot('hold name/E-7D Longitudinal Simulation/ date');
//
opts = 'ylab/Vv Cmd|Qv Cmd|Gamma Cmd|N2 Cmd/ strip';
Plot(t,[y_cmd(:,1) y_cmd(:,2) y_cmd(:,3) y_cmd(:,4)],opts),
pause;
hard;
//
opts = 'ylab/Vv Out|Qv Out|Gamma Out|N2 Out/ strip';
Plot(t,[y_cmd(:,5) y_cmd(:,6) y_cmd(:,7) y_cmd(:,8)],opts),
pause;
hard;
//
opts = 'ylab/Vv Error|Qv Error|Gamma Error|N2 Error/ strip';
Plot(t,[y_cmd(:,9) y_cmd(:,10) y_cmd(:,11) y_cmd(:,12)],opts),
```

```
pause;
hard;
//
opts = 'ylab/Velocity|V dot|Theta|P Rate|Gamma/ strip';
Plot(t,[y_cmd(:,13) y_cmd(:,14) y_cmd(:,15) y_cmd(:,16)
y_cmd(:,17)],opts),
pause;
hard;
//
opts = 'ylab/Rotor Sp (N2)|Noz Thr (FG9)|Ej Thr (FGE)|Vent Thr (FGV)/
strip';
Plot(t,[y_cmd(:,18) y_cmd(:,19) y_cmd(:,20) y_cmd(:,21)],opts),
pause;
hard;
//
opts = 'ylab/dele|AQR|ANG79|ANG8/ strip left nodate';
Plot(t,[y_cmd(:,22) y_cmd(:,23) y_cmd(:,24) y_cmd(:,25)],opts),
//
opts = 'ylab/WF|A8|ETA|A78/ strip right noname';
Plot(t,[y_cmd(:,26) y_cmd(:,27) y_cmd(:,28) y_cmd(:,29)],opts),
pause;
hard;
//
opts = 'ylab/dele rate|AQR rate|ANG79 rate|ANG8 rate/ strip left nodate';
Plot(t,[y_cmd(:,30) y_cmd(:,31) y_cmd(:,32) y_cmd(:,33)],opts),
//
opts = 'ylab/WF rate|A8 rate|ETA rate|A78 rate/ strip right noname';
Plot(t,[y_cmd(:,34) y_cmd(:,35) y_cmd(:,36) y_cmd(:,37)],opts),
pause;
hard;
//
y_cmdr=y_cmd;
clear u_cmd u_null u_step opts sopt smenu;
Plot('reset');
//\\print/queue=ps_chaos/delete matplot.ps;*
display('*** All Plots Complete! ***')
```

8.1.3.3 Closed–Loop Singular Values of Centralized Controller Plus Plant

MATRIX$_X$® "EXEC" file for weighted closed–loop singular values (STOVL_CSVCL.MTX):

```
//  Solves for weighted closed loop system singular values
//
// *** Requires S_HEW (NS_HEW) Matrix from H-infinity Design ***
//
exist('s_hew');
display('*** Order of Controller (NSK) ***'); nsk,
Plot('hold name/E-7D Longitudinal Model/ date');
Plot('hold title/WEIGHTED CLOSED LOOP SYSTEM SINGULAR VALUES/');
//
//  Compute Singular Values for Closed Loop System
//
      display('*** Computing (All Inputs/All Out) Singular Values ***');...
      [omega,sv_hew]=svplot(s_hew,ns_hew,.01,100.);
      msv_hew = 10**(sv_hew(:,1)/20.);
      pause;
```

```
//
      display('*** Computing (Zcmd In/Error Out) Singular Values ***');...
      s_hew1 = s_hew(1:60,1:60);
      [omega,sv_hew1]=svplot(s_hew1,ns_hew,.01,100.);
      msv_hew1 = 10**(sv_hew1(:,1)/20.);
      pause;
//
      display('*** Computing (Zcmd In/Z Out) Singular Values ***'),...
      s_hew2 = [s_hew(1:56,1:60);s_hew(61:64,1:60)];
      [omega,sv_hew2]=svplot(s_hew2,ns_hew,.01,100.);
      msv_hew2 = 10**(sv_hew2(:,1)/20.);
      pause;
//
      display('*** Computing (Zcmd In/u Out) Singular Values ***'),...
      s_hew3 = [s_hew(1:56,1:60);s_hew(65:72,1:60)];
      [omega,sv_hew3]=svplot(s_hew3,ns_hew,.01,100.);
      msv_hew3 = 10**(sv_hew3(:,1)/20.);
      pause;
//
      display('*** Computing (Zcmd In/u-dot Out) Singular Values ***'),...
      s_hew4 = [s_hew(1:56,1:60);s_hew(73:80,1:60)];
      [omega,sv_hew4]=svplot(s_hew4,ns_hew,.01,100.);
      msv_hew4 = 10**(sv_hew4(:,1)/20.);
      pause;
//
//  Composite Plots
//
Plot('hold grid5 xlab/Frequency, rps/ color=1 ymax=20.');
Plot('hold ylab/Singular Value Magnitude/');
//
Plot('hold legend/max(H)|max(e)|max(z)|max(u)|max(u-dot)/');
Plot(omega,[msv_hew msv_hew1 msv_hew2 msv_hew3 msv_hew4],'logx logy');
pause;
clear msv* sv* omega s_hew1 s_hew2 s_hew3 s_hew4;
```

8.2 MATRIX$_X$® Code – Subcontroller Partitioning

8.2.1 Engine Subcontroller Partitioning

MATRIX$_X$® "EXEC" file for engine subcontroller partitioning (STOVL_EERED.MTX):

```
//
//  Partition Reduced Centralized Controller for Engine Subcontroller
//
// Inputs:
//           SKRRS    - NSKRR x NSKRR Scaled Reduced Order Controller Matrix
//           NSKRR    - Order of Controller Matrix (SKRR)
//           SCALEIN  - Input Scaling for Centralized Controller
//           SCALEOUT - Output Scaling for Centralized Controller
//
// Outputs:
//           SKEER    - NSKEER x NSKEER Red. Engine Subcontroller Matrix
//           NSKEER   - Order of Red. Engine Subcontroller Matrix (SKEER)
```

```
//           SKEEF    - NSKEEF x NSKEEF Full Engine Subcontroller Matrix
//           NSKEEF   - Order of Red. Engine Subcontroller Matrix (SKEEF)
//
resize('sstack',300000);
//
//  Isolate Engine Subcontroller
//
skee1 = [skrrs(:,1:nskrr) skrrs(:,nskrr+4) skrrs(:,nskrr+10:nskrr+11)];
skee2 = [skee1(1:nskrr,:); skee1(nskrr+5:nskrr+8,:)];
nskeef = nskrr;
skeefs = skee2;
//
//  Analyze Full Order Engine Subcontroller
//
Plot('hold name/E-7D Longitudinal Engine Controller/ date');
[omega,sv_keef] = svplot(skeefs,nskrr,.01,100);
mxsv_keef = 10**(sv_keef(:,1)/20.);
mnsv_keef = 10**(sv_keef(:,3)/20.);
pause;
//
//  Rescale Full Order Engine Subcontroller
//
skeef1 = [skeefs(:,1:nskeef),...
          skeefs(:,nskeef+1)*inv(scalein(4,4)),...
          skeefs(:,nskeef+2:nskeef+3)*inv(scalein(10:11,10:11))];
skeef = [skeef1(1:nskeef,:);
         scaleout(5:8,5:8)*skeef1(nskeef+1:nskeef+4,:)];
//
//  Reduce Scaled Engine Controller
//
[skeerb,sig,t] = balance(skeefs,nskrr);
//[skeers,nskeer] = mreduce(skeerb,nskrr,[1:4]);
nskeer=4;
skeers = [skeerb(1:nskeer,1:nskeer),...
          skeerb(1:nskeer,nskrr+1:nskrr+3);...
          skeerb(nskrr+1:nskrr+4,1:nskeer),...
          skeerb(nskrr+1:nskrr+4,nskrr+1:nskrr+3)];
//
//  Analyze Reduced Order Controller
//
Plot('hold name/E-7D Longitudinal Red. Order Controller/ date');
[omega,sv_keer] = svplot(skeers,nskeer,.01,100);
mxsv_keer = 10**(sv_keer(:,1)/20.);
mnsv_keer = 10**(sv_keer(:,3)/20.);
pause;
//
//  Plot Comparison of Controller Singular Values
//
Plot('hold grid5 xlab/Frequency - rps/ color=1');
Plot('hold name/E-7D Longitudinal Engine Controller/ date');
Plot('hold ylab/Singular Value Magnitude - dB/ logx logy');
Plot(omega,[mxsv_keef mxsv_keer mnsv_keef mnsv_keer]),
Pause;
//
//  Rescale Reduced Order Engine Subcontroller
//
```

```
skeer1 = [skeers(:,1:nskeer),...
          skeers(:,nskeer+1)*inv(scalein(4,4)),...
          skeers(:,nskeer+2:nskeer+3)*inv(scalein(10:11,10:11))];
skeer = [skeer1(1:nskeer,:);
         scaleout(5:8,5:8)*skeer1(nskeer+1:nskeer+4,:)];
//
clear mxsv_keef mxsv_keer mnsv_keef mnsv_keer sv_keef sv_keer;
clear omega skee1 skee2 skeer1 skeef1 skeem skeerb sig t;
```

8.2.2 Engine–Airframe Subcontroller Design

8.2.2.1 Engine–Airframe Subcontroller Design Specification

MATRIX$_X$® "EXEC" file for engine–airframe interface design specification (STOVL_EASPEC.MTX):

```
//
//  Determine Specification for Thrust Responses
//
// Inputs:
//           SK     - NSK x NSK Controller Matrix
//           NSK    - Order of Controller Matrix (SK)
//
// Outputs:
//           XXX
//
resize('sstack',300000);
scaleint=scalein(1:3,1:3);
scaleoutea=diag([1000.0 2000.0 1000.0]);
//
//  Analyze Closed Loop with Full Order Controller
//
sim('analyze/Closed Loop');
[sclf,nsclf] = lin(0);
sclsv1t = [sclf(:,1:nsclf), sclf(:,nsclf+1:nsclf+3)*scaleint];
sclsv1  = [sclsv1t(1:nsclf,1:nsclf+3);...
           inv(scaleoutea(1,1))*sclsv1t(nsclf+19,1:nsclf+3)];
sclsv2t = [sclf(:,1:nsclf), sclf(:,nsclf+1:nsclf+3)*scaleint];
sclsv2  = [sclsv2t(1:nsclf,1:nsclf+3);...
           inv(scaleoutea(2,2))*sclsv2t(nsclf+20,1:nsclf+3)];
sclsv3t = [sclf(:,1:nsclf), sclf(:,nsclf+1:nsclf+3)*scaleint];
sclsv3  = [sclsv3t(1:nsclf,1:nsclf+3);...
           inv(scaleoutea(3,3))*sclsv3t(nsclf+21,1:nsclf+3)];
Plot('hold name/E-7D Longitudinal EA Requirements/ date');
[omega,sv_clf1] = svplot(sclsv1,nsclf,.01,100);
sv_clfm1 = 10**(sv_clf1/20);
pause;
[omega,sv_clf2] = svplot(sclsv2,nsclf,.01,100);
sv_clfm2 = 10**(sv_clf2/20);
pause
[omega,sv_clf3] = svplot(sclsv3,nsclf,.01,100);
sv_clfm3 = 10**(sv_clf3/20);
pause;
//
```

```
//  Plot Closed Loop Singular Values
//
Plot('hold grid5 xlab/Frequency - rps/ color=1');
Plot('hold legend/FG9|FGE|FGV/ date');
opts = 'ylab/Singular Value Magnitude/ logx logy';
Plot(omega,[sv_clfm1, sv_clfm2, sv_clfm3],opts),
Plot('reset');
clear sv_clf1 sv_clfm1 sclsv1 sclsv1t sv_clf2 sv_clfm2 sclsv2 sclsv2t;
clear sv_clf3 sv_clfm3 sclsv3 sclsv3t omega;
```

8.2.2.2 Engine–Airframe Subcontroller Design

MATRIX$_X$® "EXEC" file for engine–airframe interface subcontroller design (STOVL_EADES.MTX):

```
//
//  Solve H-infinity Engine-Airframe Subcontroller Design
//
define 'sg_hinf.mtx'
resize('sstack',300000);
sim('analyze/Engine Design Plant');
[se,nse] = lin(0);
nsee=[nse 15 3];
inquire gam_in 'Enter initial guess of gamma: ';
//[skea,nskea,s_eew,ns_eew] = hinf_contr(se,nsee,gam_in);
[skea,nskea,s_eew,ns_eew] = sg_hinf(se,nsee,gam_in);
[omega,sv_eew] = svplot(s_eew,ns_eew,.01,100.)
msv_eew = 10**(max(sv_eew)/20),
display('*** Rerun if gamma is much different from initial guess ***');
clear msv_eew gam_in omega sv_eew;
```

8.2.2.3 Engine–Airframe Subcontroller Order Reduction

MATRIX$_X$® "EXEC" file for engine–airframe interface subcontroller order reduction (STOVL_EARED.MTX):

```
//
//  Reduce Engine-Airframe Sub Controller
//
// Inputs:
//           SKEA     - NSKEA x NSKEA Controller Matrix
//           NSKEA    - Order of Controller Matrix (SKEA)
//
// Outputs:
//           SKEAR  - NSKEAR x NSKEAR RO Sub Controller Matrix
//           NSKEAR - Order of RO Controller Matrix (SKEAR)
//
resize('sstack',300000);
//
//  Scale Full Order Engine-Airframe Sub Controller
//
scaleinea=diag([1000 2000 1000]);
skea1 = [skea(:,1:nskea) skea(:,nskea+1:nskea+3)*scaleinea];
skeas = [skea1(1:nskea,:);...
         inv(scaleout(5:8,5:8))*skea1(nskea+1:nskea+4,:)];
```

```
//
//  Analyze Full Order Engine-Airframe Sub Controller
//
Plot('hold name/E-7D E-A Full Order Sub Controller/ date');
[omega,sv_keaf] = svplot(skeas,nskea,.01,100);
mxsv_keaf = 10**(sv_keaf(:,1)/20.);
mnsv_keaf = 10**(sv_keaf(:,3)/20.);
pause;
//
//  Reduce Scaled Sub Controller
//
[skeabs,sigeabs,teabs] = balance(skeas,nskea);
//[skears,nskear] = mreduce(skeabs,nskea,[1:3]);
sigeabs
nskear = 4;
skears = [skeabs(1:nskear,1:nskear),...
          skeabs(1:nskear,nskea+1:nskea+3);...
          skeabs(nskea+1:nskea+4,1:nskear),...
          skeabs(nskea+1:nskea+4,nskea+1:nskea+3)];
//
//  Analyze Reduced Order Sub Controller
//
Plot('hold name/E-7D E-A Red. Order Sub Controller/ date');
[omega,sv_kear] = svplot(skears,nskear,.01,100);
mxsv_kear = 10**(sv_kear(:,1)/20.);
mnsv_kear = 10**(sv_kear(:,3)/20.);
pause;
//
//  Plot Comparison of Controller Singular Values
//
Plot('hold grid5 xlab/Frequency - rps/ color=1');
Plot('hold name/E-7D Longitudinal Controller/ date');
Plot('hold ylab/Singular Value Magnitude/ logx logy');
Plot(omega,[mxsv_keaf mxsv_kear mnsv_keaf mnsv_kear]),
hard;
Pause;
//
//  Rescale Reduced Order Controller
//
skear1 = [skears(:,1:nskear) skears(:,nskear+1:nskear+3)*inv(scaleinea)];
skear = [skear1(1:nskear,:);
         scaleout(5:8,5:8)*skear1(nskear+1:nskear+4,:)];
//
//  Analyze Closed Loop with Full Order Controller
//
sim('analyze/Engine Closed Loop');
[scleaf,nscleaf] = lin(0);
scleasv = [scleaf(1:nscleaf,1:nscleaf+3);...
           scleaf(nscleaf+4:nscleaf+6,1:nscleaf+3)];
Plot('hold name/E-7D E-A Full Order Closed Loop/ date');
[omega,sv_cleaf] = svplot(scleasv,nscleaf,.01,100);
pause;
mxsv_cleaf = 10**(sv_cleaf(:,1)/20.);
mnsv_cleaf = 10**(sv_cleaf(:,3)/20.);
//
//  Analyze Closed Loop with Reduced Order Controller
```

```
//
sim('analyze/Engine Closed Loop Red');
[sclear,nsclear] = lin(0);
scleasv = [sclear(1:nsclear,1:nsclear+3);...
           sclear(nsclear+4:nsclear+6,1:nsclear+3)];
Plot('hold name/E-7D Engine-Airframe Reduced Closed Loop/ date');
[omega,sv_clear] = svplot(scleasv,nsclear,.01,100);
mxsv_clear = 10**(sv_clear(:,1)/20.);
mnsv_clear = 10**(sv_clear(:,3)/20.);
pause;
//
//  Plot Comparison of Closed Loop Singular Values
//
Plot('hold grid5 xlab/Frequency - rps/ color=1');
Plot('hold name/E-7D Engine-Airframe Closed Loop/ date');
opts = 'ylab/Singular Value Magnitude/ logx logy';
Plot(omega,[mxsv_cleaf mxsv_clear mnsv_cleaf mnsv_clear],opts),
hard;
pause;
clear mxsv_cleaf mxsv_clear mnsv_cleaf mnsv_clear sv_cleaf sv_clear;
clear scleasv;
clear mxsv_keaf mxsv_kear mnsv_keaf mnsv_kear sv_keaf sv_kear;
clear omega skea1 skeas skear1 skeams;
clear skeabs sigeabs teabs;
```

8.2.2.4 Engine–Airframe Subcontroller Closed–Loop Singular Value Analysis

MATRIX$_X$® "EXEC" file for engine–airframe interface subcontroller closed–loop singular value analysis (STOVL_EASVCL.MTX):

```
//  Solves for weighted closed loop system singular values
//
// *** Requires S_EEW (NS_EEW) Matrix from H-infinity Design ***
//
exist('s_eew');
display('*** Order of Controller (NSKEA) ***'); nskea,
Plot('hold name/E-7D Engine-Airframe Model/ date');
Plot('hold title/WEIGHTED CLOSED LOOP SYSTEM SINGULAR VALUES/');
//
//  Compute Singular Values for Closed Loop System
//
      display('*** Computing (All Inputs/All Out) Singular Values
***');...
      [omega,sv_eew]=svplot(s_eew,ns_eew,.01,100.);
      msv_eew = 10**(sv_eew(:,1)/20.);
      pause;
//
      display('*** Computing (Zcmd In/Error Out) Singular Values ***');...
      s_eew1 = s_eew(1:35,1:35);
      [omega,sv_eew1]=svplot(s_eew1,ns_eew,.01,100.);
      msv_eew1 = 10**(sv_eew1(:,1)/20.);
      pause;
//
      display('*** Computing (Zcmd In/Z Out) Singular Values ***'),...
      s_eew2 = [s_eew(1:32,1:35);s_eew(36:39,1:35)];
      [omega,sv_eew2]=svplot(s_eew2,ns_eew,.01,100.);
```

```
      msv_eew2 = 10**(sv_eew2(:,1)/20.);
      pause;
//
      display('*** Computing (Zcmd In/u Out) Singular Values ***'),...
      s_eew3 = [s_eew(1:32,1:35);s_eew(40:43,1:35)];
      [omega,sv_eew3]=svplot(s_eew3,ns_eew,.01,100.);
      msv_eew3 = 10**(sv_eew3(:,1)/20.);
      pause;
//
      display('*** Computing (Zcmd In/u-dot Out) Singular Values ***'),...
      s_eew4 = [s_eew(1:34,1:35);s_eew(44:47,1:35)];
      [omega,sv_eew4]=svplot(s_eew4,ns_eew,.01,100.);
      msv_eew4 = 10**(sv_eew4(:,1)/20.);
      pause;
//
//  Composite Plots
//
Plot('hold grid5 xlab/Frequency, rps/ color=1 ymax=20.');
Plot('hold ylab/Singular Value Magnitude/');
//
Plot('hold legend/max(H)|max(e)|max(z)|max(u)|max(u-dot)/');
Plot(omega,[msv_eew msv_eew1 msv_eew2 msv_eew3 msv_eew4],'logx logy');
pause;
clear msv* sv* omega s_eew1 s_eew2 s_eew3 s_eew4;
Plot('reset');
//\\print/queue=ps_chaos/delete matplot.ps;*
display('*** All Plots Complete! ***')
```

8.2.2.5 Engine–Airframe Subcontroller Closed–Loop Simulation

MATRIX$_X$® "EXEC" file for engine–airframe interface subcontroller closed–loop simulation (STOVL_EALSIM.MTX):

```
//
//  Analyze and Linearize Closed Loop System
//
resize('sstack',300000);
u_null = 0*[0:1:1000]';
for i=1:101,u_step(i)=0;end,
for i=102:601,u_step(i)=1;end,
for i=602:1001,u_step(i)=0;end,
//
sim('analyze/EA_EE Closed Loop');
[sclea,nsclea] = lin(0);

//
//  Compute STEP Time Histories
//          12345678901234567890123456789
  smenu = ['     SIM MENU          ';...
           '      N2 Cmd           ';...
           '      FG9 Cmd          ';...
           '      FGE Cmd          ';...
           '      FGV Cmd          ';...
           '       EXIT            '];...
    sopt = MENU(smenu,1);...
    If  sopt = 1,...
```

```
      display('*** Running Linear Simulation ***'),...
      u_cmd = [u_step u_null u_null u_null];...
      [t,y_cmd] = lsim(sclea,nsclea,u_cmd,.01);...
    Elseif  sopt = 2,...
      display('*** Running Linear Simulation ***'),...
      u_cmd = [u_null u_step u_null u_null];...
      [t,y_cmd] = lsim(sclea,nsclea,u_cmd,.01);...
    Elseif  sopt = 3,...
      display('*** Running Linear Simulation ***'),...
      u_cmd = [u_null u_null u_step u_null];...
      [t,y_cmd] = lsim(sclea,nsclea,u_cmd,.01);...
    Elseif  sopt = 4,...
      display('*** Running Linear Simulation ***'),...
      u_cmd = [u_null u_null u_null u_step];...
      [t,y_cmd] = lsim(sclea,nsclea,u_cmd,.01);...
    End,...
//
//  Plot Results
//
Plot('hold grid5 xlab/Time (Seconds)/ color=1');
Plot('hold name/E-7D Engine-Airframe SC Simulation/ date');
//
opts = 'ylab/N2 Cmd|FG9 Cmd|FGE Cmd|FGV Cmd/ strip';
Plot(t,[y_cmd(:,1) y_cmd(:,2) y_cmd(:,3) y_cmd(:,4)],opts),
pause;
hard;
//
opts = 'ylab/N2 Out|FG9 Out|FGE Out|FGV Out/ strip';
Plot(t,[y_cmd(:,5) y_cmd(:,6) y_cmd(:,7) y_cmd(:,8)],opts),
pause;
hard;
//
opts = 'ylab/N2 Error|FG9 Error|FGE Error|FGV Error/ strip';
Plot(t,[y_cmd(:,9) y_cmd(:,10) y_cmd(:,11) y_cmd(:,12)],opts),
pause;
hard;
//
opts = 'ylab/WF|A8|ETA|A78/ strip noname';
Plot(t,[y_cmd(:,13) y_cmd(:,14) y_cmd(:,15) y_cmd(:,16)],opts),
pause;
hard;
//
opts = 'ylab/WF rate|A8 rate|ETA rate|A78 rate/ strip noname';
Plot(t,[y_cmd(:,17) y_cmd(:,18) y_cmd(:,19) y_cmd(:,20)],opts),
pause;
hard;
//
clear u_cmd y_cmd u_null u_step opts sopt smenu;
Plot('reset');
//\\print/queue=ps_chaos/delete matplot.ps;*
display('*** All Plots Complete! ***')
```

8.2.2.6 Engine–Airframe Reduced Subcontroller Closed–Loop Simulation

MATRIX$_X$® "EXEC" file for closed–loop simulation of the engine–airframe interface reduced subcontroller (STOVL_EALSIMR.MTX):

```
//
//  Analyze and Linearize Closed Loop System
//
resize('sstack',300000);
u_null = 0*[0:1:1000]';
for i=1:101,u_step(i)=0;end,
for i=102:601,u_step(i)=1;end,
for i=602:1001,u_step(i)=0;end,
//
//sim('analyze/EA_EE Closed Loop');
//[sclea,nsclea] = lin(0);
sim('analyze/EA_EE Closed Loop Red'
[sclear,nsclear] = lin(0);
//
//  Compute STEP Time Histories
//          12345678901234567890123456789
  smenu = ['     SIM MENU          ';...
           '      N2 Cmd           ';...
           '      FG9 Cmd          ';...
           '      FGE Cmd          ';...
           '      FGV Cmd          ';...
           '       EXIT            '];...
    sopt = MENU(smenu,1);...
    If  sopt = 1,...
      display('*** Running Linear Simulation ***'),...
      u_cmd = [u_step u_null u_null u_null];...
      [t,y_cmd] = lsim(sclear,nsclear,u_cmd,.01);...
    Elseif  sopt = 2,...
      display('*** Running Linear Simulation ***'),...
      u_cmd = [u_null u_step u_null u_null];...
      [t,y_cmd] = lsim(sclear,nsclear,u_cmd,.01);...
    Elseif  sopt = 3,...
      display('*** Running Linear Simulation ***'),...
      u_cmd = [u_null u_null u_step u_null];...
      [t,y_cmd] = lsim(sclear,nsclear,u_cmd,.01);...
    Elseif  sopt = 4,...
      display('*** Running Linear Simulation ***'),...
      u_cmd = [u_null u_null u_null u_step];...
      [t,y_cmd] = lsim(sclear,nsclear,u_cmd,.01);...
    End,...
//
//  Plot Results
//
Plot('hold grid5 xlab/Time (Seconds)/ color=1');
Plot('hold name/E-7D Engine-Airframe SC Simulation/ date');
//
opts = 'ylab/N2 Cmd|FG9 Cmd|FGE Cmd|FGV Cmd/ strip';
Plot(t,[y_cmd(:,1) y_cmd(:,2) y_cmd(:,3) y_cmd(:,4)],opts),
pause;
hard;
//
```

```
opts = 'ylab/N2 Out|FG9 Out|FGE Out|FGV Out/ strip';
Plot(t,[y_cmd(:,5) y_cmd(:,6) y_cmd(:,7) y_cmd(:,8)],opts),
pause;
hard;
//
opts = 'ylab/N2 Error|FG9 Error|FGE Error|FGV Error/ strip';
Plot(t,[y_cmd(:,9) y_cmd(:,10) y_cmd(:,11) y_cmd(:,12)],opts),
pause;
hard;
//
opts = 'ylab/WF|A8|ETA|A78/ strip noname';
Plot(t,[y_cmd(:,13) y_cmd(:,14) y_cmd(:,15) y_cmd(:,16)],opts),
pause;
hard;
//
opts = 'ylab/WF rate|A8 rate|ETA rate|A78 rate/ strip noname';
Plot(t,[y_cmd(:,17) y_cmd(:,18) y_cmd(:,19) y_cmd(:,20)],opts),
pause;
hard;
//
clear u_cmd y_cmd u_null u_step opts sopt smenu;
Plot('reset');
//\\print/queue=ps_chaos/delete matplot.ps;*
display('*** All Plots Complete! ***')
```

8.2.3 Airframe Subcontroller Partitioning

MATRIX$_X$® "EXEC" file for airframe subcontroller partitioning (STOVL_ACREDS.MTX):

```
//
//  Reduce Airframe Sub Controller
//
// Inputs:
//           SKRR     - NSKRR x NSKRR Reduced Centralized Controller Matrix
//           NSKRR    - Order of Controller Matrix (SKRR)
//
// Outputs:
//           SKAR  - NSKAR x NSKAR Airframe Sub Controller Matrix
//           NSKAR - Order of Airframe Sub Controller Matrix (SKAR)
//
resize('sstack',300000);
//
//  Analyze Airframe Partition with Engine Subsystem Closed
//
sim('analyze/Cent Airframe Part');
[sclaf,nsclaf] = lin(0);
//
//  Scale Airframe Partition
//
scaleina=diag([scalein(1,1) scalein(2,2) scalein(3,3) scalein(5,5)...
               scalein(6,6) scalein(7,7) scalein(8,8) scalein(9,9)]);
scaleouta=diag([scaleout(1,1) scaleout(2,2) scaleout(3,3) ...
                scaleout(4,4) 1000 2000 1000]);
sclaf1 = [sclaf(:,1:nsclaf) sclaf(:,nsclaf+1:nsclaf+8)*scaleina];
sclafs = [sclaf1(1:nsclaf,:);...
```

```
         inv(scaleouta)*sclaf1(nsclaf+1:nsclaf+7,:)];
//
Plot('hold name/E-7D Airframe Partition Closed Loop/ date');
[omega,sv_claf] = svplot(sclafs,nsclaf,.01,100);
pause;
mxsv_claf = 10**(sv_claf(:,1)/20.);
mnsv_claf = 10**(sv_claf(:,7)/20.);
//
//  Reduce Scaled Airframe Sub Controller
//
[skab,sigab,tab] = balance(sclafs,nsclaf);
//[skars,nskar] = mreduce(skab,nsclaf,[1:12]);
sigab
nskar = 12;
skars = [skab(1:nskar,1:nskar),...
         skab(1:nskar,nsclaf+1:nsclaf+8);...
         skab(nsclaf+1:nsclaf+7,1:nskar),...
         skab(nsclaf+1:nsclaf+7,nsclaf+1:nsclaf+8)];
Plot('hold name/E-7D Airframe Sub Controller/ date');
[omega,sv_kar] = svplot(skars,nskar,.01,100);
pause;
mxsv_kar = 10**(sv_kar(:,1)/20.);
mnsv_kar = 10**(sv_kar(:,7)/20.);
//
//  Plot Comparison of Scaled Airframe Subcontroller Singular Values
//
Plot('hold grid5 xlab/Frequency - rps/ color=1');
Plot('hold name/E-7D Airframe Subcontroller/ date');
opts = 'ylab/Singular Value Magnitude/ logx logy';
Plot(omega,[mxsv_claf mxsv_kar mnsv_claf mnsv_kar],opts),
pause;
Plot(omega,[sv_claf(:,1) sv_kar(:,1)],opts),
pause;
Plot(omega,[sv_claf(:,2) sv_kar(:,2)],opts),
pause;
Plot(omega,[sv_claf(:,3) sv_kar(:,3)],opts),
pause;
Plot(omega,[sv_claf(:,4) sv_kar(:,4)],opts),
pause;
Plot(omega,[sv_claf(:,5) sv_kar(:,5)],opts),
pause;
Plot(omega,[sv_claf(:,6) sv_kar(:,6)],opts),
pause;
Plot(omega,[sv_claf(:,7) sv_kar(:,7)],opts),
pause;
//
//  Rescale Reduced Order Controller
//
skar1 = [skars(:,1:nskar) skars(:,nskar+1:nskar+8)*inv(scaleina)];
skar = [skar1(1:nskar,:); scaleouta*skar1(nskar+1:nskar+7,:)];
//
clear mxsv_claf mnsv_claf sv_claf;
clear mxsv_kar mnsv_kar sv_kar;
clear sigab tab
```

8.3 MATRIX$_X$® Code – Partitioned Control Law Evaluation

MATRIX$_X$® "EXEC" file for closed–loop simulation of partitioned system (STOVL_PLSIM.MTX):

```
//
//  Analyze and Linearize Closed Loop System
//
resize('sstack',300000);
u_null = 0*[0:1:1000]';
for i=1:101,u_step(i)=0;end,
for i=102:601,u_step(i)=1;end,
for i=602:1001,u_step(i)=0;end,
//
sim('analyze/Partitioned Closed Loop');
[sclp,nsclp] = lin(0);
//
//  Compute STEP Time Histories
//          12345678901234567890123456789
  smenu = ['      SIM MENU          ';...
           '   Velocity Cmd         ';...
           '     Pitch Cmd          ';...
           '    Flight Path Cmd     ';...
           ' Engine Fan Speed Cmd   ';...
           '       EXIT             '];
    sopt = MENU(smenu,1);...
    If  sopt = 1,...
      display('*** Running Linear Simulation ***'),...
      u_cmd = [u_step u_null u_null u_null];...
      [t,y_cmd] = lsim(sclp,nsclp,u_cmd,.01);...
    Elseif  sopt = 2,...
      display('*** Running Linear Simulation ***'),...
      u_cmd = [u_null u_step u_null u_null];...
      [t,y_cmd] = lsim(sclp,nsclp,u_cmd,.01);...
    Elseif  sopt = 3,...
      display('*** Running Linear Simulation ***'),...
      u_cmd = [u_null u_null u_step u_null];...
      [t,y_cmd] = lsim(sclp,nsclp,u_cmd,.01);...
    Elseif  sopt = 4,...
      display('*** Running Linear Simulation ***'),...
      u_cmd = [u_null u_null u_null u_step];...
      [t,y_cmd] = lsim(sclp,nsclp,u_cmd,.01);...
    End,...
//
//  Plot Results
//
Plot('hold grid5 xlab/Time (Seconds)/ color=1');
Plot('hold name/E-7D Longitudinal Simulation/ date');
//
opts = 'ylab/Vv Cmd|Qv Cmd|Gamma Cmd|N2 Cmd/ strip';
Plot(t,[y_cmd(:,1) y_cmd(:,2) y_cmd(:,3) y_cmd(:,4)],opts),
pause;
hard;
//
opts = 'ylab/Vv Out|Qv Out|Gamma Out|N2 Out/ strip';
```

```
Plot(t,[y_cmd(:,5) y_cmd(:,6) y_cmd(:,7) y_cmd(:,8)],opts),
pause;
hard;
//
opts = 'ylab/Vv Error|Qv Error|Gamma Error|N2 Error/ strip';
Plot(t,[y_cmd(:,9) y_cmd(:,10) y_cmd(:,11) y_cmd(:,12)],opts),
pause;
hard;
//
opts = 'ylab/Velocity|V dot|Theta|P Rate|Gamma/ strip';
Plot(t,[y_cmd(:,13) y_cmd(:,14) y_cmd(:,15) y_cmd(:,16)
y_cmd(:,17)],opts),
pause;
hard;
//
opts = 'ylab/Rotor Sp (N2)|Noz Thr (FG9)|Ej Thr (FGE)|Vent Thr (FGV)/
strip';
Plot(t,[y_cmd(:,18) y_cmd(:,19) y_cmd(:,20) y_cmd(:,21)],opts),
pause;
hard;
//
opts = 'ylab/dele|AQR|ANG79|ANG8/ strip left nodate';
Plot(t,[y_cmd(:,22) y_cmd(:,23) y_cmd(:,24) y_cmd(:,25)],opts),
//
opts = 'ylab/WF|A8|ETA|A78/ strip right noname';
Plot(t,[y_cmd(:,26) y_cmd(:,27) y_cmd(:,28) y_cmd(:,29)],opts),
pause;
hard;
//
opts = 'ylab/dele rate|AQR rate|ANG79 rate|ANG8 rate/ strip left nodate';
Plot(t,[y_cmd(:,30) y_cmd(:,31) y_cmd(:,32) y_cmd(:,33)],opts),
//
opts = 'ylab/WF rate|A8 rate|ETA rate|A78 rate/ strip right noname';
Plot(t,[y_cmd(:,34) y_cmd(:,35) y_cmd(:,36) y_cmd(:,37)],opts),
pause;
hard;
//
clear u_cmd y_cmd u_null u_step opts sopt smenu;
Plot('reset');
//\\print/queue=ps_chaos/delete matplot.ps;*
display('*** All Plots Complete! ***')
```

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (*Leave blank*)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	April 1996	Final Contractor Report

4. TITLE AND SUBTITLE

An Assessment of IMPAC—Integrated Methodology for Propulsion and Airframe Controls

5. FUNDING NUMBERS

WU–505–62–50
C–NAS1–19000

6. AUTHOR(S)

G.P. Walker, E.A. Wagner, and D.S. Bodden

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Lockheed Martin Tactical Aircraft Systems
Fort Worth, Texas

8. PERFORMING ORGANIZATION REPORT NUMBER

E–10137

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

National Aeronautics and Space Administration
Lewis Research Center
Cleveland, Ohio 44135–3191

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

NASA CR–198460

11. SUPPLEMENTARY NOTES

This report prepared for and managed by Lewis Research Center as a task of Langley Contract NAS1-19000. Project Manager, Sanjay Garg, Instrumentation and Control Technology Division, NASA Lewis Research Center, organization code 2530, (216) 433–2355.

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Unclassified - Unlimited
Subject Categories 63 and 08

This publication is available from the NASA Center for AeroSpace Information, (301) 621–0390.

12b. DISTRIBUTION CODE

13. ABSTRACT (*Maximum 200 words*)

This report documents the work done under a NASA sponsored contract to transition to industry technologies developed under the NASA Lewis Research Center IMPAC (Integrated Methodology for Propulsion and Airframe Control) program. The critical steps in IMPAC are exercised on an example integrated flight/propulsion control design for linear airframe/engine models of a conceptual STOVL (Short Take-Off and Vertical Landing) aircraft, and MATRIXX® executive files to implement each step are developed. The results from the example study are analyzed and lessons learned are listed along with recommendations that will improve the application of each design step. The end product of this research is a set of software requirements for developing a user-friendly control design tool which will automate the steps in the IMPAC methodology. Prototypes for a graphical user interface (GUI) are sketched to specify how the tool will interact with the user and it is recommended to build the tool around existing computer aided control design software packages.

14. SUBJECT TERMS

Feedback control; Propulsion control; Flight control; Robustness; Control theory

15. NUMBER OF PAGES

90

16. PRICE CODE

A05

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	